DEFENDING AGAINST ADVERSARIAL ATTACKS IN ELECTRIC POWER SYSTEMS: A MACHINE LEARNING APPROACH

By

JUN JIANG

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Engineering and Computer Science, Vancouver

July 2019

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of JUN JIANG find it satisfactory and recommend that it be accepted.

Xinghui Zhao, Ph.D., Chair

Scott Wallace, Ph.D.

Xuecheng Zhang, Ph.D.

ACKNOWLEDGMENTS

I would like to first express my gratitude to my advisor, Prof. Xinghui Zhao, who has been giving me continuous guidance and support for the past two years. Without you, I would not have had the privilege to start the journey and made it this far.

I would like to acknowledge Dr. Scott Wallace who also led me through the way. You are always inspiring and it was a pleasure to work with you. I would also like to appreciate Prof. Xuechen Zhang's willingness to serve on my committee.

In addition, I would like to thank my parents for their unlimited love. You are always there for me. Moreover, there are my friends, Priya, Suman, Long and Bao, who have made my past years bright and colorful.

Last but not least, this work cannot be done without the Department of Energy / Bonneville Power Administration for their generous support through the Technology Innovation Program (TIP# 377) and for providing the PMU data used in this study.

DEFENDING AGAINST ADVERSARIAL ATTACKS IN ELECTRIC POWER SYSTEMS: A MACHINE LEARNING APPROACH

Abstract

by Jun Jiang, M.S.
Washington State University
July 2019

Chair: Xinghui Zhao

Phasor measurement units (PMUs) provide high-fidelity situational awareness of electric power grid operations. PMU data are used in real-time to inform wide area state estimation, monitor area control error, and event detection. As PMU data becomes more reliable, these devices are finding roles within control systems such as demand response programs and early fault detection systems. As with other cyber physical systems, maintaining data integrity and security are significant challenges for power system operators. In this thesis, we present a comprehensive study of multiple machine learning techniques for detecting malicious data injection within PMU data streams. Our results show that both SVM and ANN are generally effective in detecting spoofed data using signal correlations, and TensorFlow, the open source machine learning library, demonstrates potential for distributing the training workload and achieving higher performance. We also developed a competent second-wise LSTM model based on raw signals which increases the detection granularity to seconds. We expect these results to shed light on future work of adopting machine learning and data analytics techniques in the electric power industry.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Over the past decade, smart grid technology has become an emerging and fast-growing field within both research and industry. The fundamental concept of the smart grid is to enable and enhance wide-area monitoring, control and protection of power systems by leveraging the advances in modern sensing, communication and information technologies. A core device of the smart grid is phasor measurement unit, i.e., PMU, invented by Phadke and Thorp in the 1980s [1, 2, 3]. These devices, once deployed on the electric power grid, can provide near real-time measurements of Steinmetz's current and voltage phasors, which represent the current status of the electric grid. These measurements from widely-distributed geographical locations are synchronized with a precise clock using the global positioning system (GPS); each PMU data point has a precise time stamp aligned to a common time reference [4]. This time stamp allows PMU data from disparate locations to be synchronized, thereby providing a precise and comprehensive view of the entire grid.

Because of the enhanced monitoring capability enabled by PMUs, these devices have been widely adopted by electric utilities, balancing authorities and transmission operators. From 2009 to 2014, PMU deployment in the U.S. increased from 200 PMUs [5] to approximately 1700 [6]. However, along with the value these devices bring to the power grid, they also

introduce new challenges. The volume of data PMUs generate presents challenges for the traditional workflow of grid operations. Specifically, PMU sampling and recording rate ranges from 10-60 samples per second for a 60Hz system [7]. This is much higher than conventional monitoring technologies such as supervisory control and data acquisition (SCADA) [8], which only sample once every two to four seconds. As a result, the volume of data to be stored, retrieved, processed, and analyzed is significantly larger than that of conventional systems. For instance, the Bonneville Power Administration's PMU newtork generates about 1.5 terabytes of data per month. This number is increasing as new PMUs are added to the system. Besides the big data challenge, data security and integrity are also critical concerns. Data integrity can be compromised due to various causes, such as data drops, clock drifts, or injection of deceptive data signals. These types of deterioration can disrupt PMU data, affect control operations and ultimately lead to major problems, such as cascading failures.

To address these challenges, research developed within the fields of big data analytics and machine learning can be applied to efficiently process and analyze PMU data, as well as detect any data disruption when it happens [9]. In this thesis, we present our work on evaluating three widely used machine learning methods, Support Vector Machines (SVM), Artificial Neural Networks (ANN) and Long Short-term Memory (LSTM) Networks, on detecting malicious data injections in PMU data streams. We use two datasets containing PMU data collected from real power systems. The first dataset, *BPA_Data*, was collected by PMUs from Bonneville Power Administration's wide-area monitoring system in the U.S. Pacific Northwest area. These PMUs are deployed on the 500 kV transmission network. The second dataset, *OSU_Data*, was collected from the inter-university PMU network we built with three universities in the region: Washington State University in Vancouver, Portland State University, and Oregon State University. These datasets represents PMU data from the distribution level. Using these datasets, we conducted a comprehensive evaluation of the effectiveness of SVM, ANN as well as LSTM in detecting spoofed signals with PMU data

streams.

The contributions of this thesis are multifold. First, we have identified a set of features which are robust and effective in both SVM and ANN, and for both datasets. Using these features, both SVM and ANN can identify spoofs without extensive parameter tuning. Second, to the best of our knowledge, this is the first comprehensive evaluation of multiple machine learning methods for spoof detection within PMU data streams. Third, we showed that the performance of training in ANN can be improved significantly by leveraging the techniques of distributed computing. This provides potential for future work in real-time detection of spoofed signals. Forth, we developed a second-wise LSTM model which can leverage more PMU data and achieve situational awareness of smart grids in seconds. And lastly, to the best of our knowledge, this thesis is the first to apply machine methods to PMU data at both the transmission level and the distribution level. Our results show that SVM, ANN and LSTM are all effective for detecting spoofed data within datasets. We expect these results to shed light on future work of adopting machine learning and data analytic techniques in the applicable to the electric power industry.

The organization of the rest of the thesis is as follows. In Chapter 2, we present related work in the cyber security aspect of the power grid, as well as applications of machine learning methods in this field. Chapter 3 introduces the two datasets used in this study and discusses their characteristics. In Chapter 4, we describe the methodology of our work including data processing, feature selection, and evaluation metrics. Chapter 5 presents the evaluation results of the three machine learning techniques, in terms of their performance in detection, and training timespan for SVM and ANN. Finally, Chapter 6 concludes the thesis and presents potential future directions for this work.

Chapter 2

Related Work

Cyber-security has long been a major concern for critical infrastructure, which are assets that are essential for the functioning of a society and economy [10]. Examples of critical infrastructures include electric power, natural gas, oil pipeline, and water supply systems. These facilities are monitored and controlled using Supervisory Control and Data Acquisition (SCADA) systems, in which measurement data are collected by widely-distributed remote terminal units, and delivered to a control center, i.e., the master station. In past decades, these systems, as well as their data communication channels, have been targeted by cyber attackers [11]. Cyber-security incidents involving critical infrastructure and SCADA systems are well documented [12]. These attacks, sometimes intelligently designed and executed, are difficult to detect, especially when disguised via spoofing techniques. For instance, in 2010, Stuxnet [13, 14, 15], a computer worm designed to be inflicted upon on industrial equipment, altered the setpoint speed of the drives used to control centrifuges in the Iranian nuclear facility at Natanz, causing thousands centrifuges not function as intended. During the attack, Stuxnet used spoofed data to mask its malicious activities, so that operators were not aware of the setpoint changes. Another example is a series of attacks reported by McAfee in 2011, namely 'Night Dragons'. These attacks targeted global energy and oil

firms, and exfiltrated critical data such as operational blueprints [16]. These attacks had been ongoing for more than two years before they were identified, because the attackers used a set of tools to compromise the target computers and mask their identity.

Most recently, with more information and networking technologies being introduced to critical infrastructures including power grids, the cyber security concern in these system has received an ever increasing amount of attention. Since PMUs are introduced to power grids as critical data sources, these devices and their communication channels are also under the threat of cyber-security attacks. PMUs are key devices in electric power systems, providing measurements for state estimators, initiating remedial action schema, and estimating voltage-stability margins [17]. In [18], threat potential has been demonstrated, where the difference between a PMU's receiver GPS clock offset before and after an attack is maximized [18]. In addition, the consequences of an attack on the time stamps of data collected within a smart grid wide-area network is investigated in [19]. A comprehensive survey and evaluation of the vulnerabilities of PMUs to GPS spoofing attacks can be found in [20].

Compared with SCADA systems, PMUs provide sample data operate at a much higher rate, normally at 30 or 60 samples per second, as opposed to traditional SCADA refresh rate of seconds to even minutes. As a result, the amount of data generated by PMUs is significantly larger than that of a SCADA-based system. Therefore, big data challenges are inevitably introduced to these systems, in addition to the vulnerability to cyber attacks. To address these challenges, big data and machine learning techniques may be leveraged and applied to PMU data storage and processing. A variety of machine learning techniques have been applied to analyze PMU data for the purpose of recognizing patterns or signatures of events. It has been demonstrated that both classification [21] and clustering [22] are effective methods in analyzing PMU data streams for event detection. In addition, one class learning has the potential to identify anomalies in PMU data [23].

Machine learning techniques have proven to be effective at detecting security attacks in

cyber-physical systems [24] [25], including electric power systems [26]. However, to the best of our knowledge, there is no previous work on comparing and evaluating multiple machine learning methods using a generic set of features extracted from PMU data. The features we used in this thesis are based on Pearson correlation coefficients between PMU data streams [27]. A similar type of feature has been used in [28] for spoof detection, although it focuses on Global Navigation Satellite Systems (GNSS) signals instead of PMU streams. Also, no previous work has been done in evaluating machine learning methods using real PMU data streams from both the transmission level and distribution level of the smart grid. This thesis presents our work in this direction. Furthermore, we have demonstrated the potential for achieving more efficient training performance by leveraging a distributed computing framework. This is the critical initial step to apply these methods to real-time PMU data streams for online spoof detection, which has not been addressed in previous literature.

Chapter 3

Datasets

The smart grid structure and our data collection method are shown in Figure 3.1. The structure of an electric power grid is composed of four main components: generation, transmission, distribution, and consumers. These components represent the basic workflow of a power grid. Generation refers to various power plants that generate power. The transmission network consists power lines and substations responsible for transmitting the electricity from its place of generation to the distribution level. These lines and substations operate at a high voltage level (e.g., 500 kV), in order to minimize large voltage drops and I^2R losses within transmission line conductors. The distribution network represents the final stage in the delivery of electric power, distributing electricity from the transmission system to individual consumers. Distribution substations convert high voltage power from the transmission system to a medium-range voltage for distribution (under 100kV) using step-down transformers. Primary distribution lines carry this medium voltage power to distribution transformers located near customer premises. Distribution transformers lower the voltage further to utilization voltages used by lighting, industrial equipment and household appliances.

Within an electrical grid, PMUs may be deployed at both the transmission and distribu-

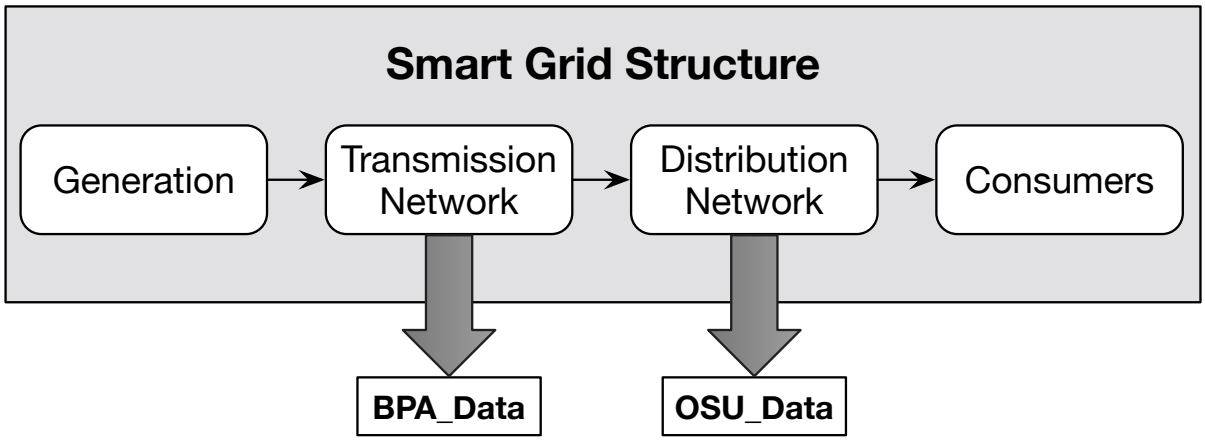


Figure 3.1: Data Collection on the Smart Grid

tion levels to enhance situation awareness. These devices, as well as their communications network, may become the targets of cyber attacks. In order to develop an effective approach for detecting such attacks, it is essential to evaluate it on both the transmission level and the distribution level. Therefore, we have collected and analyzed PMU data from both levels for this study. As illustrated in Figure 3.1, *BPA_Data* came from a transmission network, and *OSU_Data* came from a distribution network.

3.1 Transmission Level Dataset

BPA_Data, is a dataset provided by Bonneville Power Administration, one of the first transmission operators to implement a comprehensive adoption of synchrophasors in their wide-area monitoring system. This dataset contains data streams collected by 10 PMUs, from BPA's 500 kV PMU network. Note that there are more PMUs deployed in BPA's transmission network. We chose 10 PMUs to use in this study for the following reasons. First, based on historical cyber-security incidents, these attacks usually have one specific target, either a device or a network channel. Therefore, to mimic these attacks and evaluate the

spoof detection approaches, using data from a small set of PMUs is sufficient. Second, being able to detect cyber attacks using only local information from nearby PMUs is critical. This enables efficient detection, so that the same technology can be easily scaled up to the whole system, using a divide-and-conquer approach. The 10 PMUs selected for this study are electrically-close to each other, based on our calculation of the electrical distance, which has been shown in [29] to be a useful representation of power system connectivity. Third, selecting 10 PMUs keeps the dataset at the same scale as our distribution level dataset, for the purpose of fair comparison.

3.2 Distribution Level Dataset

OSU_Data, is a dataset collected by 7 PMUs with our self-deployed, inter-university PMU network. This dataset represent PMU data on the distribution level of the grid. Our research PMU network consists of seven PMUs, one each at the Washington State University-Vancouver (WSU-V) and Portland State University (PSU) campuses, with the remaining five placed at multiple locations across the Oregon State University (OSU) campus in Corvallis. The PMUs provide monitoring at the utilization level (120/208 V), with the exception of two PMUs at OSU, which monitor a 4kV and a 20kV distribution substation, respectively. All PMUs monitor three phase services. In our distribution level PMU network, all the PMUs report data at 60 samples per second to a Phasor Data Concentrator (PDC) located at the OSU campus. The data management scheme on the PDC emulates real PDC setups. A local archive stores 60 days of data on the PDC. The files are also archived to an another server for permanent storage. In addition to utilizing commercially available software from SEL, which stores data in CSV and SynchroWAVe formats, the PDC uses PDAT recording software that is currently deployed at Bonneville Power Administration. All of these file-types are archived for permanent storage so that we may reliably access them from a variety

of software tools.

These two datasets together provide data representing both the high-voltage transmission level network, as well as the medium-voltage distribution level network. Using these dataset, we can evaluate spoof detection techniques in a comprehensive manner. The main characteristics and collection dates of both datasets used in SVM and ANN are summarized in Table 3.1.

Dataset	**Type**	**# PMUs**	**Date Collected**	**Duration**
BPA_Data	Transmission	10	2013-06-26	14min
OSU_Data	Distribution	7	2017-06-03	14min

Table 3.1: Datasets used in SVM and ANN

For our second-wise LSTM model, more data were collected and are summarized in Table 3.2.

Dataset	**Type**	**# PMUs**	**Date Collected**	**Duration**
BPA_Data	Transmission	10	2016-08-10	9h
OSU_Data	Distribution	7	2017-12-05	6h

Table 3.2: Datasets used in LSTM

Chapter 4

Spoof Detection Using Machine Learning

One lesson learned from major cyber-security incidents, including Stuxnet and Night Dragon, is that the attackers often mask their malicious activities via spoofing. In other words, they inject spoofed signals to the system so that the cyber attacks that are performed can be disguised. These spoofed signals are designed in a way that the system cannot easily identify the difference. This presents an additional challenge for detecting the underlying attacks. To mimic these effects, we developed a spoofing strategy to generate spoofed signals. In this section, we present our spoofing strategy, and our methodology, including feature extraction, machine learning techniques, and evaluation metrics.

4.1 Spoofing Strategy

Data spoofing is often used in cyber attacks to mask malicious activities. Sophisticated spoofs are similar to normal data, making it difficult for the system or the operators to identify such attacks. Previous work has been done to mimic this effect and develop various

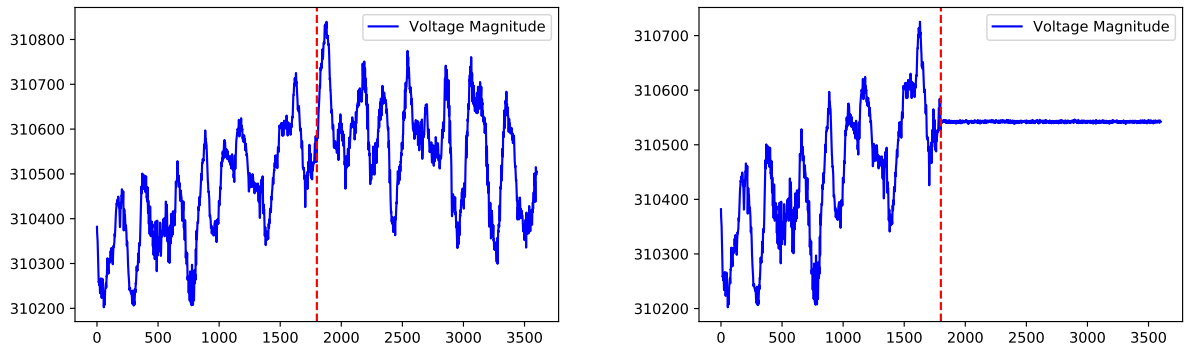
spoofing strategies for PMU data streams [27].

In our work, we examine three kinds of spoof strategies. Our spoofing strategies aim to prevent easy detection and are designed in consideration of the following constraints: (1) spoofs should provide historically-reasonable values for the target signals; (2) the onset of a spoof should not create a discontinuity that is, a signal, s, sampled immediately before the spoof, s_{t-1}, and at the start of the spoof, s_t, should show a change in value that is consistent with historical movement of that signal; (3) spoofed data should only require knowledge of the local signals, and not, for example, knowledge of signals at other PMUs. Below, we describe the three strategies employed for this study: *Repeated-Last-Value,* Mirroring and *Time-Dilation.*

The Repeated-Last-Value (RLV) spoof represents the strategy consistent with the constraints outlined above. This spoof requires knowledge of the true signal values immediately prior to the spoof onset, and repeats these values for the duration of the spoof. That is, assuming a spoof begins at time t, the signal s is given a value $s_{t+i} = s_{t-1} \forall i > 0$. Figure 4.1 shows the effect of the RLV spoof. Here, we show the voltage magnitude signal measured by one of the PMUs at BPA in a 1-minute window. Figure 4.1a shows the original data stream we collected. The red dotted line indicates the 30th second in time, i.e., 1800 cycles, where the spoofed signal will be injected. Figure 4.1b shows the signal after the RLV spoof has been applied. The signal on the right side of the dotted line is replaced by the spoofed signal, i.e., a repetition of the last value in the 29th second with Gaussian noise added for calculating correlations explained in the next chapter.

The main short-coming of the RLV spoof is that it provides a constant value signal. Although the signal can clearly be guaranteed to remain within historic range, and will produce no discontinuity at the spoof onset, the distribution of values during the spoofed period will clearly be abnormal.

The mirroring spoof improves upon the shortcoming of the RLV. To stage this attack,



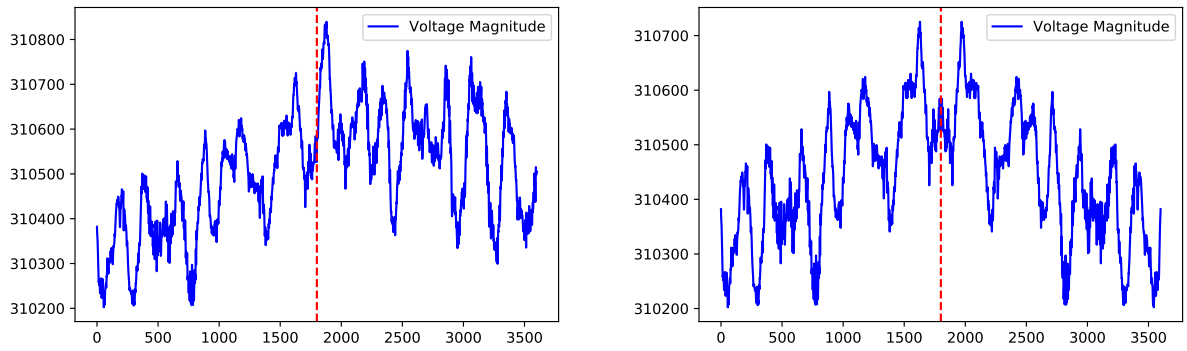
(a) Original Signal

(b) Spoofed Signal

Figure 4.1: Data Spoofing Strategy: RLV

the adversary records the target signal for a period of time u prior to the spoof, and then when the spoof begins at time t, sets $s_{t+i} = s_{t-i}$ for $i = 0...u$. However, mirroring imposes an immediate sign change on the signals first derivative ($s'_{t-1} = -s'_{t+1}$). Thus, mirroring produces an inflection point in the signal data when the spoof begins, violating our second constraint. Figure 4.2 shows the effect of the mirroring spoof. Figure 4.2b shows the signal after the mirroring spoof has been applied. The signal on the right side of the dotted line is replaced by the spoofed signal, i.e., a reverse replay of the data in the first 30 seconds. Note that replaying the signal in reverse guarantees signal continuity for all parameters at the instance spoofing is initiated.

The mirroring spoof is a simple but powerful spoofing strategy. The spoof is derived from the real signal, carrying the same characteristics, such as noise level, frequency, etc. The high similarity between real signal and the spoofed signal presents challenges for the spoof detection methods. In addition, the mirroring spoof is easy to calculate, and since it is only based on an historic data stream, it is possible to generate as much spoofing as needed to mask the real data. The mirroring spoof is used in our work to evaluate the spoof detection approaches. Specifically, we perform mirroring spoofing on one of the PMUs in



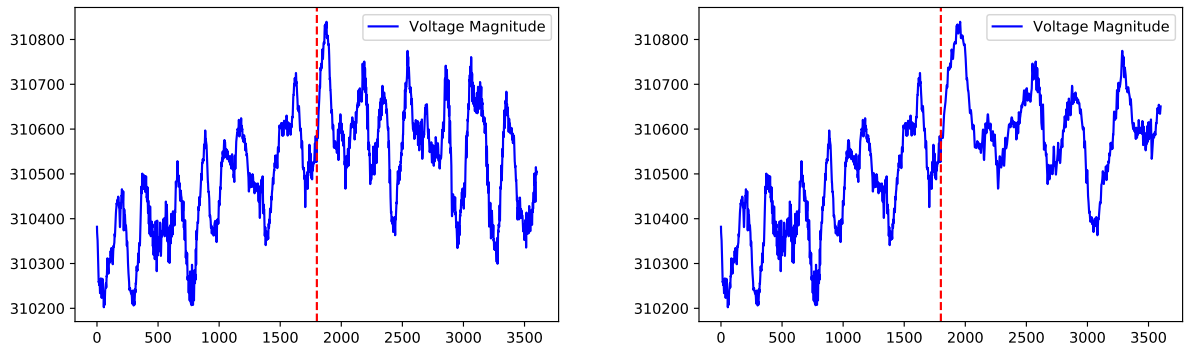
(a) Original Signal

(b) Spoofed Signal

Figure 4.2: Data Spoofing Strategy: Mirroring

each datasets and use the spoofed data to evaluate the effectiveness of each approach in accurately detecting the spoofed signals.

The Time-Dilation spoof plays recorded data back at a slower than normal rate. A time dilation of $2x$ records data for n cycles, but plays these back over $2n$ cycles. Like mirroring, time-dilation upholds the three constraints while providing a historically-reasonable distribution of signal values. Unlike mirroring, however, time-dilation also improves continuity of the signals first derivative. Time-dilation preserves the sign of the signals derivative, but impacts the derivatives magnitude proportionally to the amount of dilation. In addition, unlike RLV and Mirroring, which provide an immediate window for malicious action when the spoof begins, time-dilation requires that the attacker continues to monitor, record, and resample live signal data even as the spoof begins. As a result, the attackers window for malicious action occurs sometime after the spoof has begun, instead of at the onset. Figure 4.3 shows the effect of the dilation spoof. Figure 4.3b shows the signal after the dilation spoof has been applied. The signal on the right side of the dotted line is replaced by the spoofed signal, i.e., a $2x$ slower replay of the data in the first 30 seconds.



(a) Original Signal

(b) Spoofed Signal

Figure 4.3: Data Spoofing Strategy: Dilation

4.2 Feature Extraction

Feature extraction is a critical step in applying machine learning techniques to solve a problem. When performing analysis of complex data, one of the major problems stems from the number of variables/features involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause a classification algorithm to overfit to training samples and generalize poorly to new samples. Feature extraction aims for constructing combinations of the features while still describing the data with sufficient accuracy. In our work, one of the main objectives is to develop a generic set of features from PMU data, which can be used by multiple machine learning algorithms in detecting spoofed signals.

PMUs measure phasors of line voltages and line currents for all voltages (A, B, C) and currents (A, B, C, N). From these are derived a number of other parameters, including magnitude and phase angle for the positive, negative and zero sequence voltages and currents; frequency; and rate of change of frequency (ROCOF); among others [7]. After examining all the PMU signals, we found that intra-PMU parameters (i.e., correlation between different signals from the same PMU) are usually weakly correlated, yet the inter-PMU parameters

(i.e., the same signal from different PMUs) are often highly correlated, especially when the PMUs are electrically close to each other. These observations indicate that the correlations between PMU signals have great potentials to serve as features to construct the models in machine learning techniques.

To quantify the degree of correlation between PMU parameters, we use *Pearson Correlation Coefficient (PCC)* as the metric. The PCC of two data streams $X(x_1, x_2, ..., x_n)$ and $Y(y_1, y_2, ..., y_n)$ can be calculated as shown in Equation 4.1.

$$PCC = \frac{\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{Y})^2}} \tag{4.1}$$

Specifically, given PMUs numbered $1, 2 \ldots, p$ we develop $\binom{p}{2}$ vectors of correlation values between a specific signal for every pair of PMUs $i < j$. This is repeated for eight signals: positive sequence voltage magnitude $|V_+|$, negative sequence voltage magnitude $|V_-|$, zero sequence voltage magnitude $|V_0|$, positive sequence phase angle ϕ_+, negative sequence phase angle ϕ_-, zero sequence phase angle ϕ_0, frequency f, and the rate of change of frequency, ROCOF. After carefully examining the correlation data for these eight signals, we have the following observations. First, correlation vectors $r(|V_+|)$, $r(\phi_+)$ and $r(f)$ are good candidates for detecting spoofing attacks, as these consistently exhibit moderate to high correlation values over wide ranges of time. The $r(\phi_+)$ correlation values are exceptionally high, near 1.0 under normal circumstances. Second, ROCOF correlation between PMUs is very poor, likely due to the fact that it is the second derivative of the positive sequence phase angle, and hence more susceptible to noise. Third, correlations on other signals, including $r(|V_-|)$, $r(|V_0|)$, $r(\phi_-)$ and $r(\phi_0)$, do not exhibit consistent moderate correlation. Therefore, they may or may not add values to the learning process.

Based on our observations, we have chosen five correlation features to use in the machine learning techniques. These include the three strongly correlated features, $r(|V_+|)$, $r(\phi_+)$ and

$r(f)$, and two moderately correlated features, $r(\phi_-)$ and $r(\phi_0)$. These features are used in different learning algorithms to evaluate their effectiveness. Note that these correlation values fluctuate with time, since the correlation is performed using data windows of a fixed length. In this work, we chose a fixed window of 5-second (300 cycles).

4.3 Support Vector Machines

Using the correlation features we derived from the raw PMU data streams, we carried out a comprehensive evaluation on two widely used machine learning techniques, Support Vector Machine (SVM) [30], and Artificial Neural Networks (ANN) [31].

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary classifier. Specifically, an SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. Here in our case, we use the two-class SVM to learn a relationship that differentiates spoofed PMU signals from the normal signals. We leverage the Python library sci-kit learn for a Support Vector Machine implementation based on libsvm [32, 33].

4.4 Artificial Neural Networks

Similar to SVM, Artificial Neural Network (ANN) is also a widely used technique for supervised learning. It is a machine learning model inspired by the biological nervous system.

This technique has been widely applied in the fields of computer vision, speech recognition, anomaly detection, etc. However, to the best of our knowledge, it has not been evaluated using power systems' data, in the context of spoof detection.

An ANN is based on a collection of connected units called artificial neurons. Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal and then send it to the downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent.

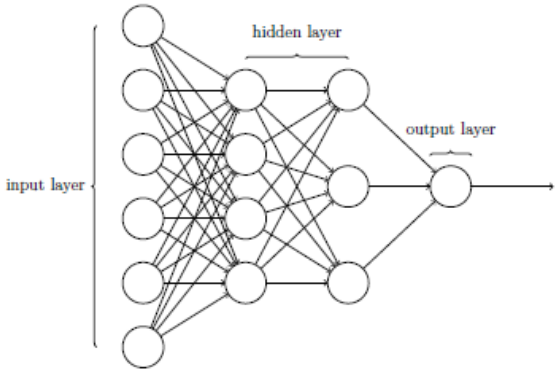


Figure 4.4: A Typical ANN Structure

Figure 4.4 demonstrates a typical ANN structure. As we can see, neurons are organized in layers. Layers are made up of a number of interconnected nodes which contain an activation function. Data features are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done via a system of weighted connections. The hidden layers then link to an output layer where the results of the learning are made available to the users. In our work, we built an ANN which has two hidden layers, each with 10 neurons. Activation functions in hidden layers are both *TanH* which computes

hyperbilic tangent. The activation function of the output layer is *Softmax*, which computes softmax activations.

4.5 Long Short Term Memory Networks

Even though ANNs have shown strong capability in a number of regression and classification tasks, the process ANNs focus on is relatively static. In other words, the outputs of a ANN only depend on the current inputs. It doesn't take into consideration the sequential relationship of inputs. Human brains, however, are capable of dealing with sequential data. When we are reading a sentence, we can infer the accurate meaning of a poly-semantic word by putting it into its own context and understand the sentence as a whole. When we are listening to music, we are able to appreciate the pleasure of melody because musical notes are arranged in accordance with the principles of music. That is to say, the perception by human brains is not only associated with the current inputs but also previous inputs. The kind of neural networks that simulate this specific brain behavior is called recurrent neural networks. This feature of RNNs makes it promising in our spoofing detection tasks since our raw data from the smart grid is time series and our primary spoofing strategy, mirroring, is sequential. It's hard to tell if it is spoofed by just looking at a single isolated sample. But given its previous signals, a mirroring pattern is likely to be detected. In addition, when using SVM and ANN, the model takes as input carefully selected and calculated features which requires complicated pre-processing. In practice, we want to leave out the real-time processing which can increase the overhead of training. RNNs, as a deep learning method, can learn directly from raw data without the need of pre-processing. We will introduce basic RNNs in the beginning of this section and then an improved version with so-called Long Short Term Memory (LSTM) cells.

4.5.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are popular models in dealing with sequential information such as time series data in Natural Language Processing (NLP) tasks. Sequence models such as RNNs are capable of capturing temporal information so that knowledge learned in a previous time period can be carried along and assist future learning process. Denny Britz has an excellent explanation on RNNs in his blog [34]. A typical RNN looks like Figure 4.5 [35] below.

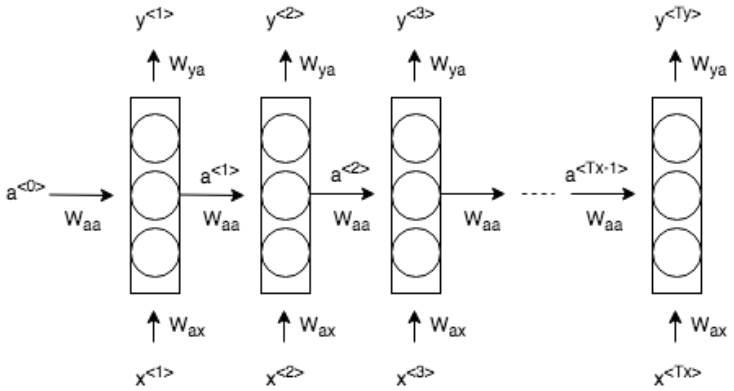


Figure 4.5: The unfolding in time of an RNN model

Figure 4.5 demonstrates the unfolded structure of an RNN model with the parameters involved in its forward propagation. The unfolded structure shows clearly what an RNN model does to an input sequence. Basically, it consumes the elements of the sequence one by one and make predictions along the way or in the end depending on your tasks. It also presents the key difference between RNNs and basic ANNs. That is, the output at current timestamp is depended on both current inputs and results from previous computations. RNNs have a "memory" or so-called hidden state to keep track of information that has been calculated so far and make use of that information to assist decision making at current time step. By unfolding an RNN model like Figure 4.5, we are able to better understand how it deals with a complete sequence. First, the number of layers of an unfolded RNN equals to

the length of the input sentence. For example, if the input sentence has 9 words in an NLP task, the RNN would be unfolded into 9 layers, one layer for each word. Second, it performs the same task for each word. That's why the same parameters are labeled for each layer in the diagram. f

4.5.2 Notations and Formulas

To best describe and understand how RNNs work. Let's first start by defining a notation that we will use to build up a RNN model [35]. The motivating example looks at such a NLP task where the input is a sentence, for example, "Harry Potter and Hermione Granger invented a new spell", and we want our sequence model to identify which word is part of a name. In the previous case, the output is expected to be [1, 1, 0, 1, 1, 0, 0, 0, 0]. This kind of problem is called Named Entity Recognition and can be used by search engines to index people's names mentioned in news feeds.

We will take the following notations to build an RNN model. Let $X^{(i)<t>}$ be the t^{th} element of the input sequence in the i^{th} training example, $T_x(i)$ be the length of the input sequence of the i^{th} training example, $Y_x^{<t>}$ be the be the t^{th} element of the output sequence in the i^{th} training example, and $T_y(i)$ be the length of the output sequence of the i^{th} training example. For the same training example, the length of the output sequence $T_y(i)$ can be different from the length of the input sequence $T_x(i)$. It just happens that $T_x(i)$ equals $T_y(i)$ in our motivating example.

Assume now we have a sequence of time series data as input such as a sentence, how will an RNN model process it? The forward propagation in an RNN model consists of Equation 4.2 and 4.3 where a^t denotes the activation and $\hat{y}^{<t>}$ denotes the output at step t.

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{4.2}$$

$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{4.3}$$

The first activation function g_1 can usually be $tanh$ or $ReLu$. The second activation function g_2 can usually be $sigmoid$ when it's a binary classification task or $softmax$ when it's a k-way classification task.

$$W_a = [W_{aa} \quad W_{ax}] \tag{4.4}$$

To further simply the formulas, we can create an augmented coefficient matrix W_a from the two coefficient matrices W_{aa} and W_{ax} as in Equation 4.4. Then we have Equation 4.5 and 4.6.

$$a^{<t>} = g_1(W_a[a^{<t-1>}, x^{<t>}] + b_a) \tag{4.5}$$

$$\hat{y}^{<t>} = g_2(W_y a^{<t>} + b_y) \tag{4.6}$$

Now that we have seen the unfolding in time of an RNN model and all relevant formulas, it also helps our understanding by looking into the inside of an RNN cell as simply illustrated in Figure 4.6 with a single tanh neural network layer. It conveys the same idea we have been repeating all the way that the activation at step t depends on both the current input and previous activation.

4.5.3 Long Short Term Memory(LSTM)

Even though RNNs were introduced with the appealing feature of preserving what was learned from previous steps to the present step, basic RNNs usually don't perform well in tasks which have long-term dependencies. For example, in an NLP task where we need to predict which personal pronouns, e.g. he or she, to use as the next word in a sentence. To decide this, we may need contexts from way further back when the sentence's subject was

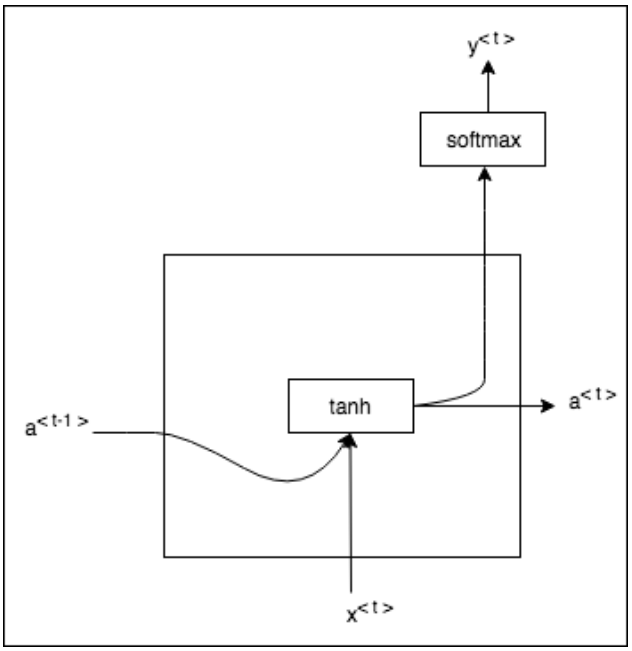


Figure 4.6: LSTM chain structure

first brought up. The distance of long-term dependency between the useful information and the place where it is needed can be very large and thus it can cause vanishing gradients in the training process of RNNs. Long Short Term Memory (LSTM) networks, which are usually just called LSTMs, were then proposed by Hochreiter and Schmidhuber [36] to address problems with long-term dependencies.

LSTMs are a special and advanced type of RNNs which have a more complicated module structure as illustrated in Figure 4.7[37]. Multiple such modules are repeated and chained together to process a sequence of inputs. Instead of a single tanh layer as in basic RNNs, there are three more layers, as well as a memory cell state which is the core idea behind LSTMs. As you can see from the top of Figure 4.7, a horizontal line runs through LSTM modules, with the previous memory cell state $c^{<t-1>}$ as input and the current memory cell state $c^{<t>}$ as output. The memory cell state conveys what have been learned so far along the way to assist the learning process at current step. To protect and control the memory cell state, LSTMs use three gates, namely the "forget" gate, the "update" gate and the

"output"gate. The "forget" gate Γ_f decides what old information to throw away from the

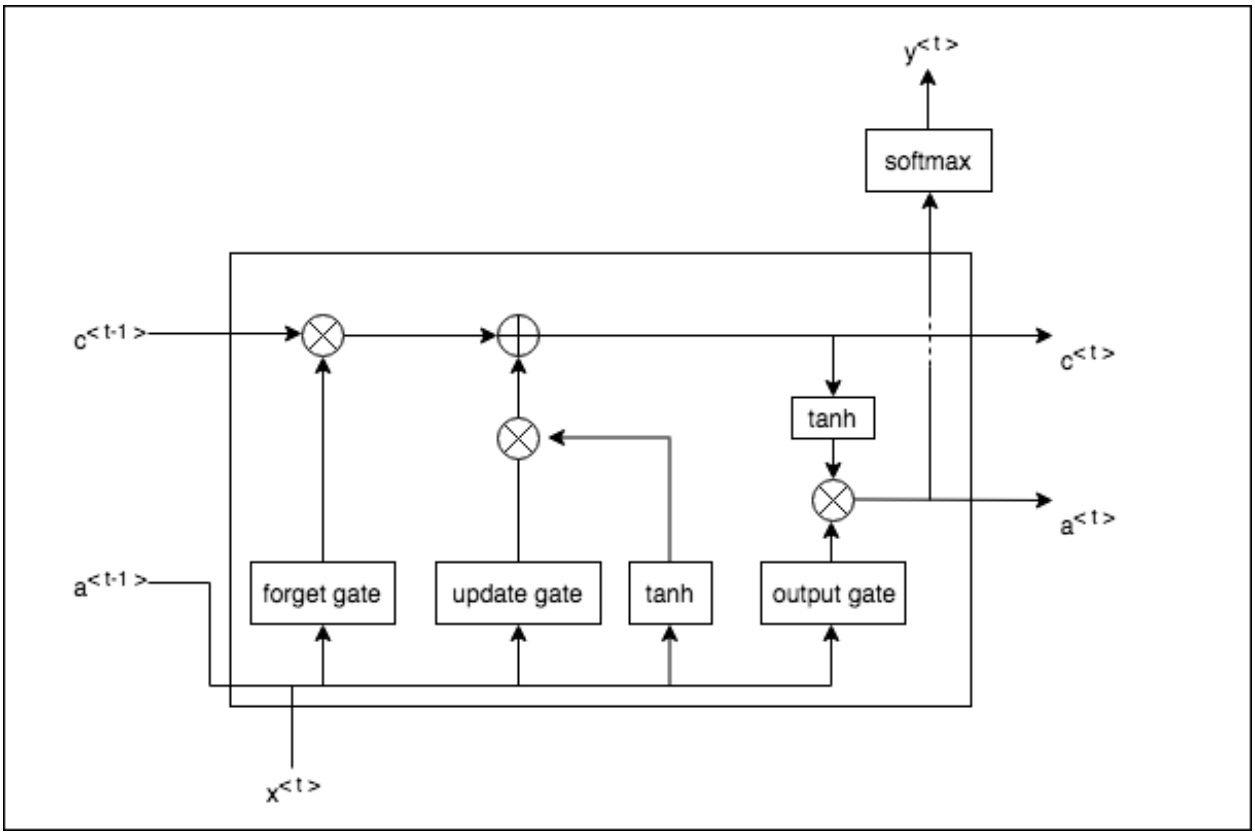


Figure 4.7: LSTM chain structure

memory cell state. It takes consideration the activation $a^{<t-1>}$ from the previous step and the current input $x^{<t>}$, and outputs a vector filled with values between 0 and 1 which has the same dimension as $c^{<t-1>}$. According to Equation 4.7, a sigmoid function σ is used to scale the values between 0 and 1 where 0 means "do not keep any information" and 1 means "keep the current memory cell state unchanged". In our previous pronoun example, this is where we forget the old subject when a new subject is brought up.

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{4.7}$$

Next, we need to decide what new information to embed into the memory cell state with help of the "update" gate Γ_u. Similar to Equation 4.7, Equation 4.8 also makes use of a sigmoid function to get a vector of values between 0 and 1 which decides which memory cell values to update. Equation 4.9 represents a tanh layer like what we have in basic RNNs and it consumes the activation $a^{<t-1>}$ from the previous step and the current input $x^{<t>}$ to construct a candidate memory cell state $\widetilde{c}^{<t>}$. In our previous pronoun example, this is where we want to add the new subject information to the memory cell state.

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{4.8}$$

$$\widetilde{c}^{<t>} = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{4.9}$$

Equation 4.10 has two terms. The first term $\Gamma_u * \widetilde{c}^{<t>}$ stands for what new information we want to embed into the memory cell state and the second term $\Gamma_f * c^{<t-1>}$ stands for what we want the memory cell state to be after forgetting outdated information. By combining these two terms, we finally complete updating the memory cell state. In the case of our pronoun example, this is where we actually forget the old subject and add information about the new subject.

$$c^{<t>} = \Gamma_u * \widetilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{4.10}$$

Now that we have updated the memory cell state, we also have to decide the output of this module. Here in Equation 4.11 we have an "output" gate similar to the "forget" gate and the "update" gate. Equation 4.12 uses the "output" gate to calculate the activation so that it can be passed to the next step or used to generate the output at this step.

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{4.11}$$

$$a^{<t>} = \Gamma_o * tanh(c^{<t>}) \tag{4.12}$$

4.5.4 Second-wise LSTMs

In applying LSTMs to implement a second-wise model, we can view each second of data as a two dimensional picture such as in the digit recognition task. Take the BPA data for example. For each of the 10 PMUs, we are considering 7 signals, consisting of frequency, magnitudes of all three phases (A, B, C) and angles of all three phases (A, B, C). Also a PMU reports data at the rate of 60 samples per second. Therefore, for each second, we can view the data as a 60 * 70 dataframe (or picture). Figure 4.8 shows how our data look if we take 5 hours (18000 seconds) of BPA data. In our LSTM model, we have 3 LSTM layers each with 256 nodes.

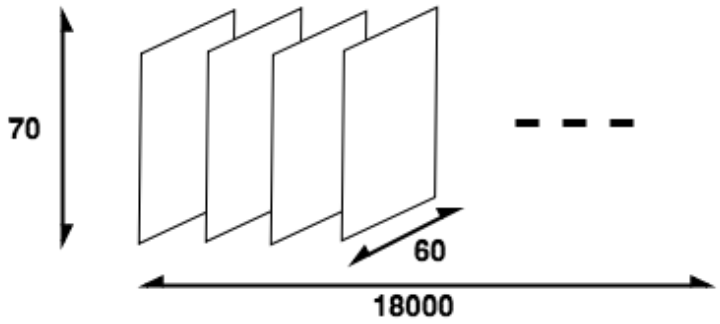


Figure 4.8: Second-wise LSTMs Data Preparation

The three primary advantages of a second-wise LSTM model are as follows:

- LSTM leverages more PMU data to grasp long-term situational awareness of the smart grid. LSTM increases the monitoring granularity to each second instead of each cycle which is 1/60 second in SVM and ANN. This extends LSTM's monitoring ability to cover longer periods.

- LSTM requires fewer possible re-training times. As the time goes, SVM or ANN models can become stale thus will not be able to reflect the current pattern of the grid. More frequent re-training processes are needed to get the model updated. Due to the reason above, LSTM requires much fewer re-training times.

- LSTM avoids the complicated and verbose data preprocessing. Since LSTM is a deep learning algorithm, we can just give as input the raw signals without any preprocessing. The model will learn the latent pattern of the data through the training process by itself.

4.6 Evaluation Metrics

To evaluate the effectiveness of the two machine learning techniques on detecting spoofed signals in PMU data, we carried out a number of experiments and took measurements on multiple metrics. Below, we describe the performance metrics we use in this study.

- **Accuracy**: measures the classifier's performance over all examples in terms of identifying examples correctly. It is calculated as the number of correct predictions, i.e., true positives (normal examples identified as such) plus true negatives (spoofed examples identified as such), divided by the total number of examples. Accuracy ranges from 0% to 100% with an ideal classifier measuring 100% accuracy.

- **Sensitivity**: measures the ability to correctly detect spoofed signals, and is calculated as the number of true positives (spoofed examples identified as such) divided by the number of total positives (the total number of spoofed examples which is the sum of true positives and false negatives). Sensitivity ranges from 0% to 100% with an ideal classifier measuring 100% sensitivity.

- **Precision:** measures how many of the positively classified were relevant and is calculated as the number of true positives (spoofed examples identified as such) divided by the number of detected spoofs (false positives plus true positives). Precision ranges from 0% to 100% with an ideal classifier measuring 100% precision.

- **Specificity:** measures the ability to correctly identify normal signals. It is calculated as the number of true negatives (normal examples identified as such) divided by the number of total negatives (the total number of normal examples which is the sum of true negatives and false positives). Specificity ranges from 0% to 100% with an ideal classifier measuring 100% specificity.

- **F1:** measures performance as a single value when classes are not equally prevalent. It is the harmonic mean of Sensitivity and Precision. F1 score ranges from 0.0 to 1.0, higher values are better.

- **False Discovery Rate (FDR):** measures the propensity to spuriously identify a spoof. This value is calculated as the number of false positives (normal examples identified as spoofs) divided by the number of detected spoofs (false positives plus true positives). False Discovery Rate is equivalent to (1-Precision). FDR ranges from 0% to 100%; an ideal classifier has 0% FDR.

Chapter 5

Evaluation

Experiments have been carried out using both the BPA and OSU datasets to evaluate the effectiveness of SVM, ANN and LSTM in terms of detecting spoofed signals. In addition, we also investigated the potential of distributing the training task to increase the computational performance. In this section, we present the process of preparing training and testing datasets, as well as the experimental results.

5.1 Training and Testing Data

First, we will introduce the two datasets used for SVM and ANN. For both the BPA and OSU datasets, we prepare the training and testing examples as follows. First, we choose 14 independent minutes of normal data from the dataset. We then apply the mirroring spoof procedure to the last 30 seconds of one selected PMU signal on each of 14 different minutes of data. Finally, we calculate the pair-wise correlation features, as described in Chapter 4.2. This approach generates roughly $2{\cdot}10^6$ examples from the 14 minutes of data and the 45 PMU pairs (i.e., 10 PMUs) in the BPA dataset, and roughly $1 \cdot 10^6$ examples from the 14 minutes of data and the 21 PMU pairs (i.e., 7 PMUs) in the OSU dataset. Examples are "Spoofed"

in the last half of each minute if i is the spoofed PMU, and are "Normal" otherwise. Given the 14 minutes of data, we use 11 minutes (roughly $1.6 \cdot 10^6$ examples for BPA data and $8 \cdot 10^5$ examples for OSU data) for training, and 3 minutes (roughly $4.5 \cdot 10^5$ examples for BPA data and roughly $2.1 \cdot 10^5$ examples for OSU data) for testing. During training, all correlations features are standardized (normalized to 0 mean and standard deviation of 1). The normalization transforms from the training features are saved so they can later be used to transform testing data prior to being classified. Note that for the parameter selection in SVM, we have performed a grid search as follows. We first split the 11 training minutes into two sets (8 and 3 minutes respectively) and performed a grid search over the C, γ parameter space by training on the former set and testing on the later. We observed high performance (F1 $>$ .95) across a wide range of parameter settings for both datsets. Thus, in subsequent sections, our results are obtained using the same set of parameters, $C = 1.0$, $\gamma = 0.2$.

Second, we use much larger datasets for the LSTM model. In total, 9 hours of BPA data and 6 hours of OSU data are collected, each with 1 hour of data for testing and the rest for training. In the case of SVM and ANN where data preprocessing is required, 11 minutes of BPA data can generate roughly $1 \cdot 10^6$ examples and the number of examples increases in a factorial manner with the number of PMUs involved since we need the combination of PMUs to calculate correlations. However, in the case of LSTM, 1 hour data has 3600 examples and the size of training data can increase linearly if more PMUs are involved.

5.2 Spoof Detection Performance

For each of our datasets, we train a two-class SVM and an ANN using 11 minutes of data, and then test its detection performance using the other 3 minutes of data. For each experiment, we measure the performance metrics described in Chapter 4.6. The results for each experiment are shown in Table 5.1, Table 5.2 and Table 5.3.

Metrics	SVM_BPA	NN_BPA	SVM_OSU	NN_OSU
Accuracy	99.13%	99.57%	93.75%	94.26%
Sensitivity	96.00%	97.74%	71.33%	71.21%
Precision	96.06%	97.59%	92.60%	96.21%
Specificity	99.52%	98.99%	98.73%	99.38%
F1	0.960	0.977	0.806	0.818
FDR	3.94%	2.41%	7.40%	3.79%

Table 5.1: Spoof Detection Performance Comparison for Mirroring

Metrics	SVM_BPA	NN_BPA	SVM_OSU	NN_OSU
Accuracy	98.96%	99.56%	95.83%	97.21%
Sensitivity	93.36%	96.01%	83.50%	85.44%
Precision	98.10%	99.99%	93.07%	99.11%
Specificity	99.78%	100.00%	98.62%	99.83%
F1	0.957	0.980	0.880	0.918
FDR	1.90%	0.01%	6.93%	0.89%

Table 5.2: Spoof Detection Performance Comparison for RLV

For all cases, we achieved high overall accuracy (ranging from 89.77% to 99.56%) and specificity (ranging from 97.62% to 100.00%), indicating that both techniques are effective in correctly identifying normal signals across the transmission level and distribution level. As for precision and FDR, both methods perform better on the BPA data. This is because the transmission level dataset has less noise compared to the distribution level dataset. For most cases, the sensitivity is relatively lower than other metrics, especially for the dilation spoof, ranging from 49.57% to 97.05%. This metric measures the percentage of spoofed signals being correctly identified by the learning algorithms. This is attributed to the following two reasons. First, since there is only one PMU being spoofed, there are more normal examples than the spoofed ones (3.5-4 times in both datasets). Therefore the algorithms learn better in terms of identifying normal data. Second, the calculation of these metrics is based on individual examples/time points. Each cycle is an example. If a cycle is within the later 30 seconds of the spoofed minute, it is labeled as spoofed. However, the features representing

Metrics	SVM_BPA	NN_BPA	SVM_OSU	NN_OSU
Accuracy	98.90%	98.42%	89.77%	90.34%
Sensitivity	91.78%	91.59%	54.45%	49.57%
Precision	97.97%	93.82%	83.54%	94.78%
Specificity	99.77%	99.26%	97.62%	99.39%
F1	0.948	0.927	0.659	0.651
FDR	2.03%	6.18%	16.5%	5.22%

Table 5.3: Spoof Detection Performance Comparison for Dilation

this example are the correlations from a 300-cycle time window before this time point, which means that some of the spoofed examples have correlation features composed of mainly non-spoofed data. This fact may also affect the sensitivity. Nonetheless, the overall performance of both techniques are high on both datasets, indicating that the features we use are generic across transmission level and distribution level.

It is worth noting that a key feature of neural networks is an iterative learning process in which examples are presented to the network one at a time, and the weights associated with the input values are adjusted each time. Typically, this process is repeated for multiple iterations. In our experiments, we trained the network over 100 such iterations. However, there is a potential that the neural network achieves good performance with even less training time. To this end, we have measured the performance metrics for mirroring after each time the network is trained, and the results are shown in Figure 5.1. For the larger BPA dataset, our neural network achieves near optimal performance after being trained 600 times, while for the smaller OSU dataset, this number is decreased to approximately 150. This indicates that for a system with smaller numbers of PMUs, the neural network may achieve optimal performance with a much smaller number of epochs.

5.3 Improvement by Minute-wise Normalization

To further improve the detection performance, we employ a minute-wise normalization method for the training data. Previously, we normalize the training data using its global mean and standard deviation. Instead of that, we normalize each minute of training data using its own mean and standard deviation. By doing this, we expect the training data to be more representative and thus improve the performance. Table 5.4, Table 5.5 and Table 5.6 confirm the improvement.

Metrics	**SVM_BPA**	**NN_BPA**	**SVM_OSU**	**NN_OSU**
Accuracy	99.79%	99.85%	98.63%	98.79%
Sensitivity	98.25%	99.21%	92.33%	93.89%
Precision	99.91%	99.97%	99.47%	99.49%
Specificity	99.93%	99.96%	99.86%	99.88%
F1	0.991	0.996	0.958	0.966
FDR	0.09%	0.03%	0.53%	0.51%

Table 5.4: Spoof Detection Performance for Mirroring with Minute-wise Normalization

Metrics	**SVM_BPA**	**NN_BPA**	**SVM_OSU**	**NN_OSU**
Accuracy	99.73%	99.86%	99.34%	99.25%
Sensitivity	99.02%	99.43%	95.13%	96.71%
Precision	99.86%	99.79%	99.74%	99.63%
Specificity	99.99%	99.87%	99.58%	99.92%
F1	0.994	0.996	0.974	0.981
FDR	0.14%	0.21%	0.26%	0.37%

Table 5.5: Spoof Detection Performance for RLV with Minute-wise Normalization

5.4 Improving Computational Performance

To further improve the performance of the neural network, we have carried out experiments to evaluate the training performance of a neural network on a distributed system using Tensorflow [38], an open-source software library for machine learning. We train our neural

Metrics	SVM_BPA	NN_BPA	SVM_OSU	NN_OSU
Accuracy	99.31%	99.64%	96.97%	97.08%
Sensitivity	96.47%	98.17%	85.32%	85.44%
Precision	97hthcs.29%	98.81%	98.34%	98.62%
Specificity	99.55%	99.73%	99.27%	99.79%
F1	0.969	0.985	0.914	0.916
FDR	2.71%	1.19%	1.66%	1.38%

Table 5.6: Spoof Detection Performance for Dilation with Minute-wise Normalization

network using Tensorflow on three systems, with 1 node, 2 nodes, and 3 nodes. Each computing node has a 4-core Intel Xeon CPU @ 3.20GHz, and 8GB RAM. On each system, we vary the training time from 100 to 1000, and measure the computational performance, i.e., training time. The results are shown in Figure 5.2.

On all three systems, the training performance is linearly related to the number of trainings. When running on a distributed system, the neural network on both datasets gain significant speedup, which indicates that the network can be easily scaled by leveraging distributed resources. This is essential when applying this approach to a much larger system, or when training the algorithm for online spoof detection, which has a higher requirement on the efficiency of the training.

5.5 Second-wise LSTM Performace

According to Table 5.7, Table 5.8 and Table 5.9, in general the second-wise LSTM model achieves better performance than SVM and ANN without the new normalization and achieves similar pattern in performance to what we have in SVM and ANN when the new normalization is applied. A close scrutiny of the results can lead to the following conclusions about the LSTM model.

1. LSTM achieves generally better performance for all three spoof types using OSU data.

This demonstrates LSTM's capability in learning the spoof patterns when the data are noisy and even without data pre-processing. Such deep learning methods have stronger learning power with the use of just raw data.

2. LSTM's performance on BPA data is comparable in terms of F1 scores to what we have in SVM and ANN with the new normalization applied, except for dilation. This is because dilation has been shown to be more difficult to detect. Despite this, we have found that LSTM's performance can be improved when more data are added into training.

3. LSTM achieves comparable or even lower FDR than SVM and ANN on both BPA and OSU data. Given the fact that we wrap up each second data as a whole instead of looking at individual samples and their correlations, we ended up with much fewer number of warnings and false alarms, which can help dramatically in further inspection.

Dataset	**Accuracy**	**Sensitivity**	**Precision**	**Specificity**	**FDR**	**F1**
BPA_Data	97.28%	94.56%	100.00%	100.00%	0.00%	0.972
OSU_Data	98.33%	97.17%	99.49%	99.50%	0.51%	0.983

Table 5.7: LSTM Performance on Mirroring

Dataset	**Accuracy**	**Sensitivity**	**Precision**	**Specificity**	**FDR**	**F1**
BPA_Data	95.36%	90.94%	99.76%	99.78%	0.24%	0.951
OSU_Data	97.25%	95.28%	99.19%	99.22%	0.81%	0.972

Table 5.8: LSTM Performance on RLV

Dataset	**Accuracy**	**Sensitivity**	**Precision**	**Specificity**	**FDR**	**F1**
BPA_Data	91.50%	84.78%	97.95%	98.22%	2.05%	0.909
OSU_Data	95.08%	93.94%	96.13%	96.22%	3.87%	0.950

Table 5.9: LSTM Performance on Dilation

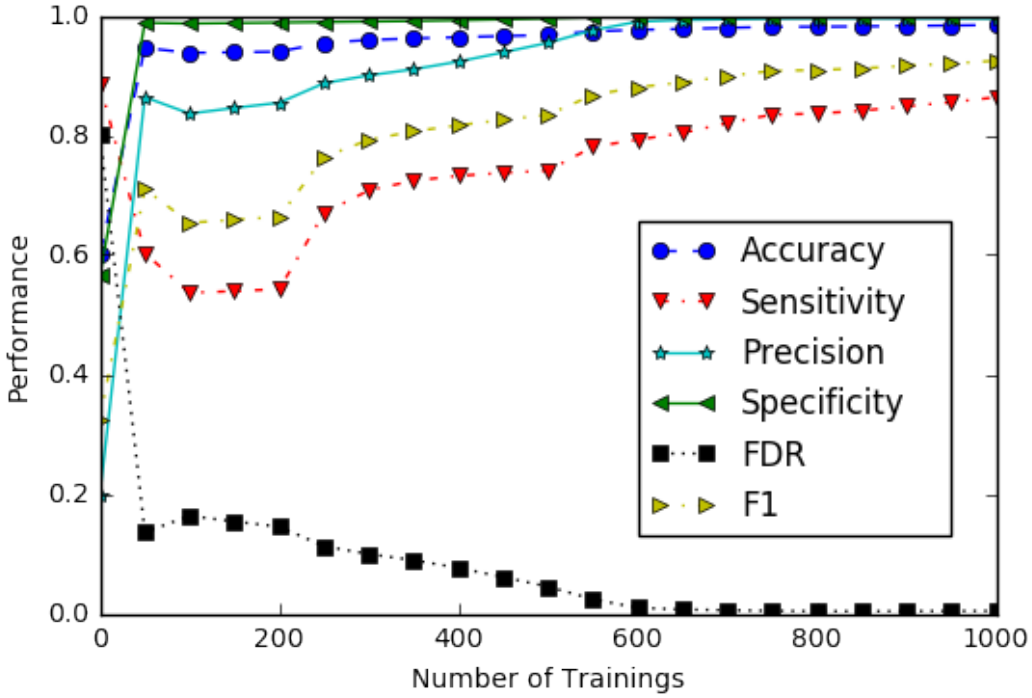
Besides the above metrics, to further demonstrate the effectiveness of our second-wise LSTM model, we also examined its training and testing time, as well as its performance on the first five seconds when the spoof starts. In terms of time consumption, on our machine with 48 Intel Xeon CPU @ 2.20GHz, it can takes up to 10 hours to train 8 hours of BPA data for around 300 epochs but it only takes around 2 seconds to test 20 minutes of data.

Dataset	**Mirroring**	**RLV**	**Dilation**
BPA_Data	2.23	2.88	4.72
OSU_Data	1.50	2.05	1.93

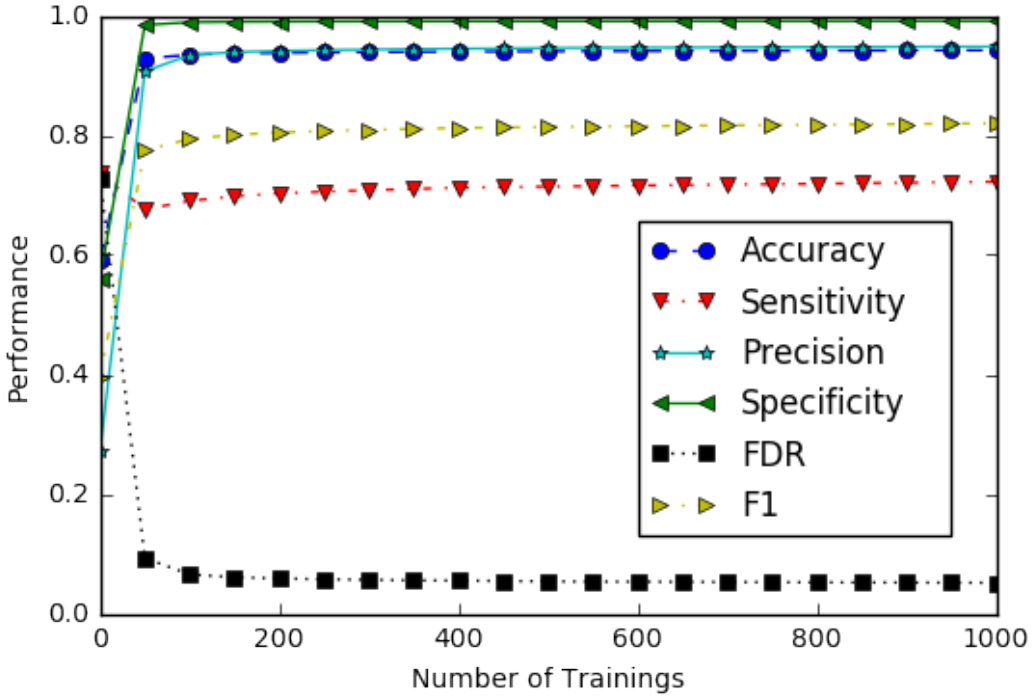
Table 5.10: LSTM Quickness

Dataset	**Mirroring**	**RLV**	**Dilation**
BPA_Data	3.95	3.62	2.85
OSU_Data	4.58	4.10	4.03

Table 5.11: LSTM Average Detected Seconds in First Five Seconds of Spoofing

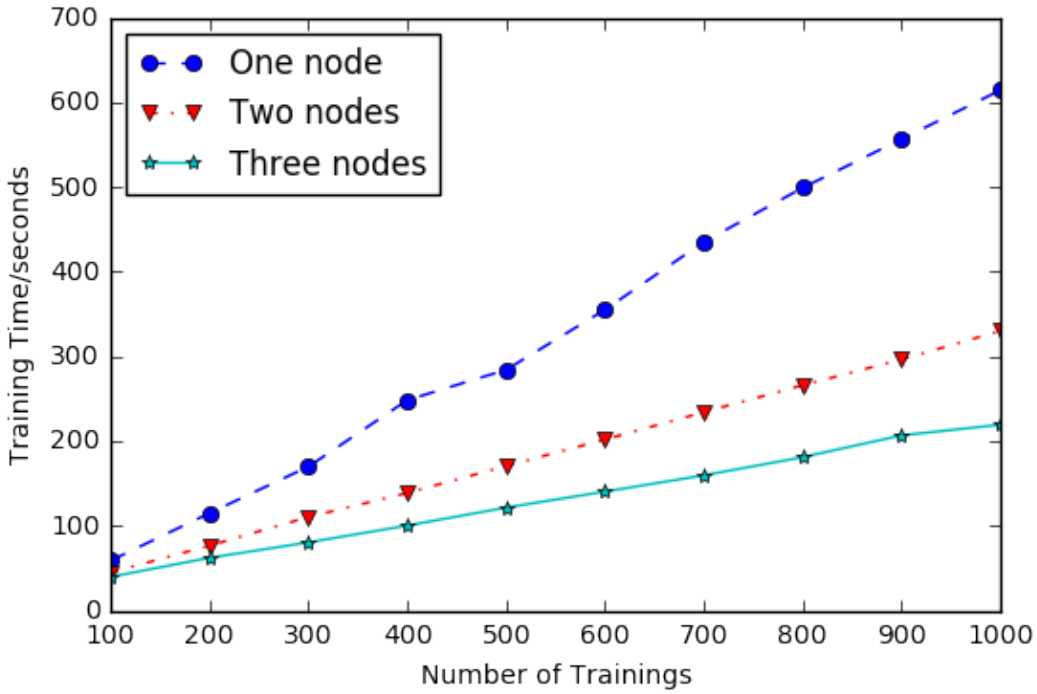


(a) BPA Data

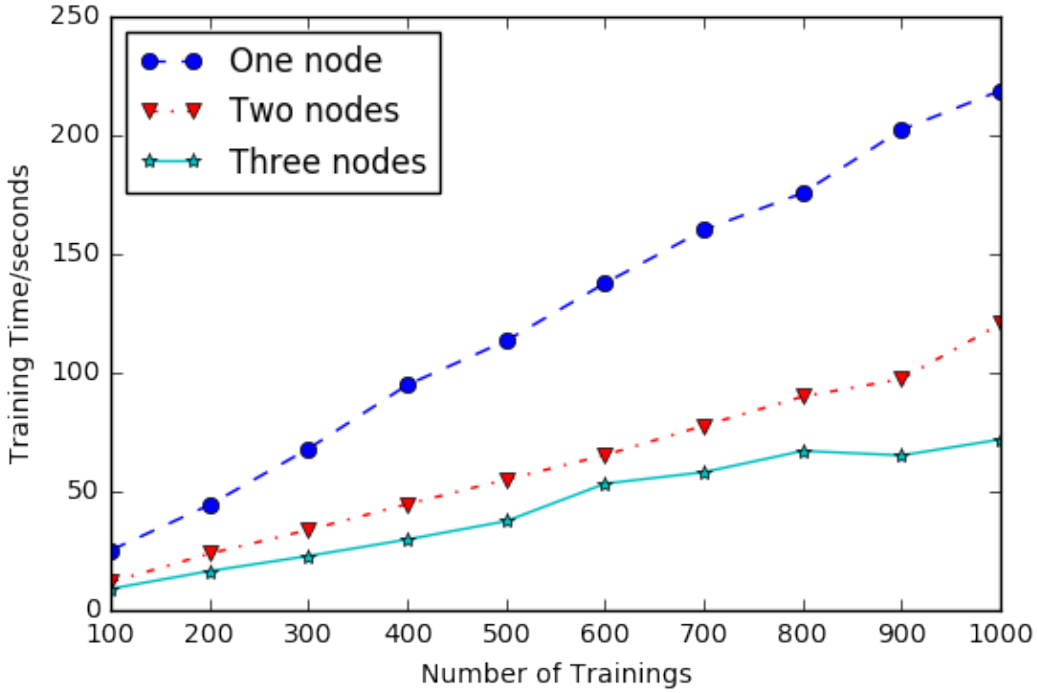


(b) OSU Data

Figure 5.1: ANN Performance Vs. Training Times

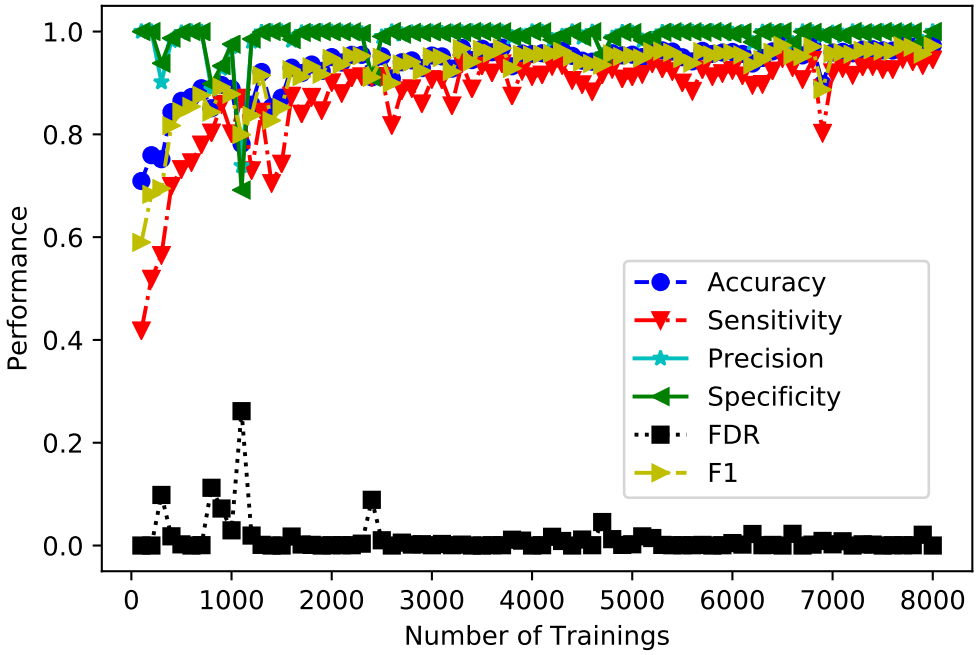


(a) BPA Data

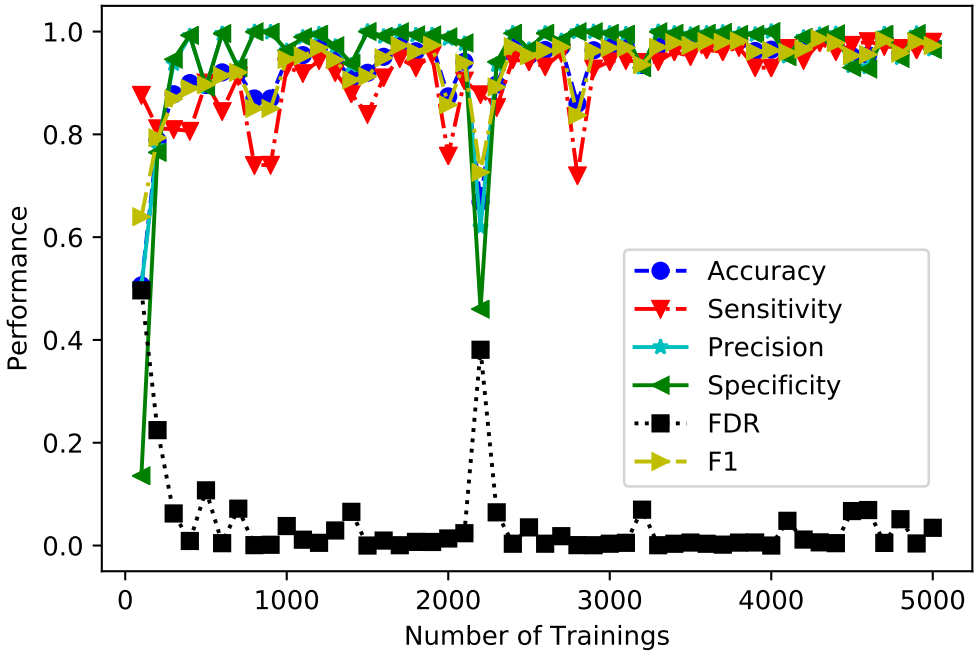


(b) OSU Data

Figure 5.2: ANN Training Performance on Distributed Systems

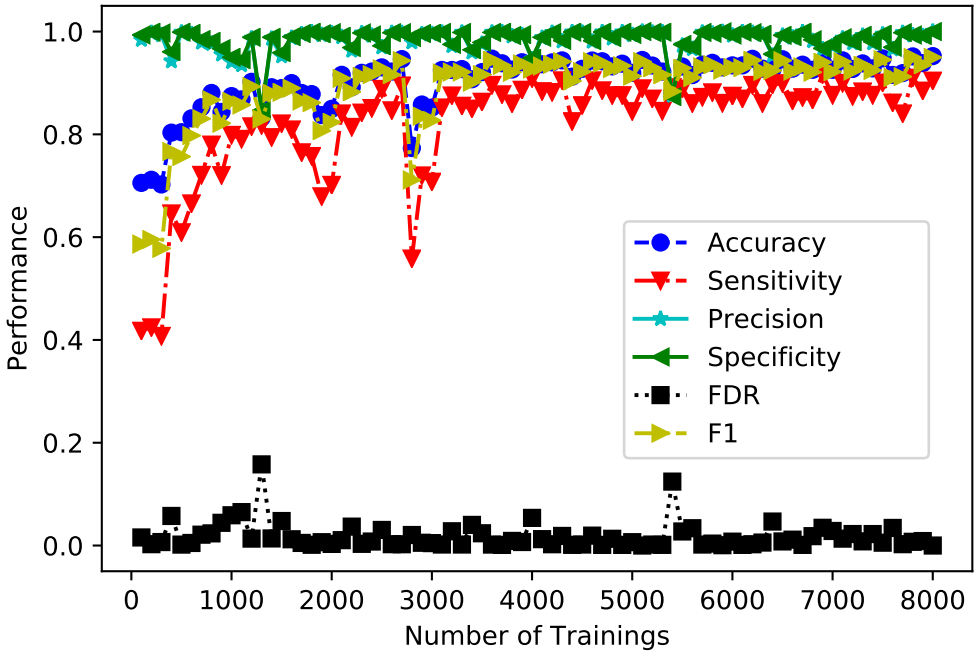


(a) BPA Data

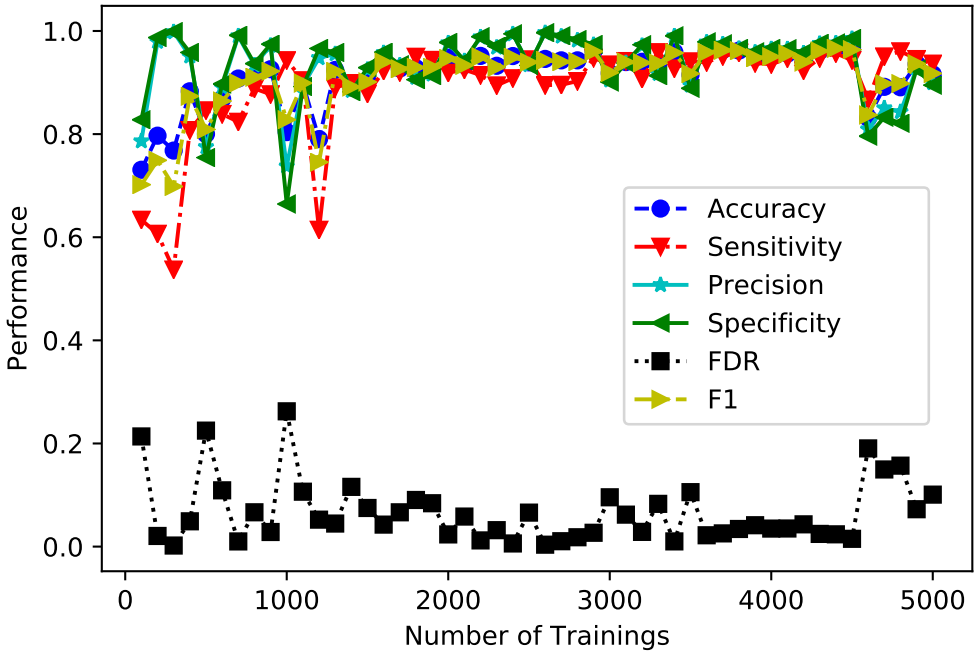


(b) OSU Data

Figure 5.3: LSTM Performance on Mirroring

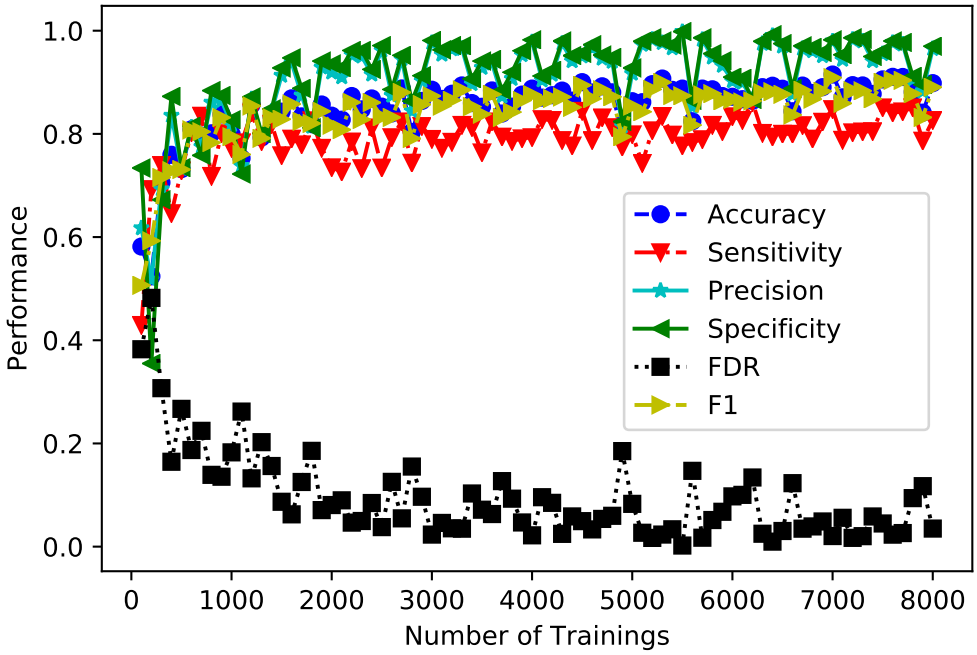


(a) BPA Data

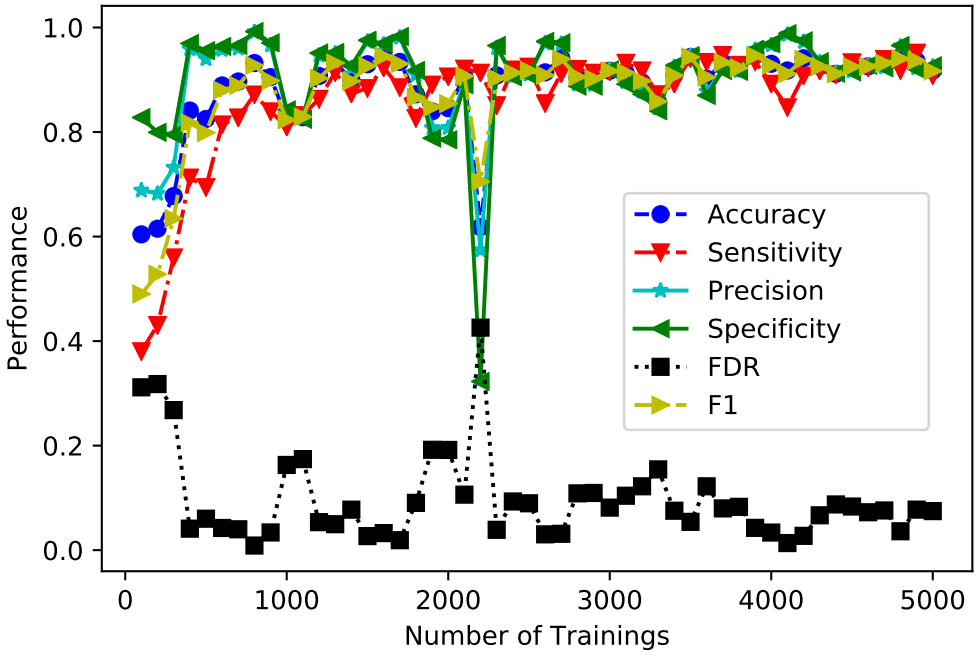


(b) OSU Data

Figure 5.4: LSTM Performance on RLV



(a) BPA Data



(b) OSU Data

Figure 5.5: LSTM Performance on Dilation

Chapter 6

Conclusion and Future Work

With information and communication technologies being integrated in modern power systems, big data and cyber security challenges become more pronounced. Historical cyber attack incidents indicate that spoofing is a common approach for disguising malicious activities. Spoofed data injected into a normal data stream make it difficult to identify the underlying attack. To address this challenge, we propose to apply machine learning techniques to PMU data for the purpose of detecting potential spoofs. Specifically, we first develop a generic set of correlation data based on features from PMU data. We then train a SVM and an ANN. To perform a comprehensive evaluation, we use two datasets, one from BPA's transmission level 500 kV network, and the other from our inter-university, distribution level PMU network. Our experimental results show that both techniques perform well after being trained using the same set of features. Moreover, the neural network approach demonstrates a good potential in achieving highly efficient training performance, via minimizing training iterations, or leveraging distributed resources. In addition, a minute-wise LSTM model is developed with the potential of simplifying the data processing and preserving good performance at the same time.

Work is on going in multiple directions. First, we will explore specificity (true negative

rate) across a large, contiguous sample. Second, we will investigate the possibilities of adjusting the performance of the spoof detection algorithm based on anticipated threat levels in response to cyber-defense intelligence. That is, we expect to be able to modify detectability at the cost of increased computational complexity or increased false positive rate in response to anticipated events. Third, we will use bi-directional LSTM to further optimize the learning by inspecting potential spoofs from both sides of the PMU data.

Bibliography

[1] A. Phadke, "Synchronized phasor measurements-a historical overview," in *IEEE PES Asia Pacific Transmission and Distribution Conf. and Exhibition*, vol. 1, Oct 2002, pp. 476–479 vol.1.

[2] C. P. Steinmetz, "Complex quantities and their use in electrical engineering," in *Proc. American Institute of Electrical Engineers*, Chicago, IL, 1893, pp. 33–74.

[3] J. De La Ree, V. Centeno, J. Thorp, and A. Phadke, "Synchronized phasor measurement applications in power systems," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 20–27, June 2010.

[4] E. O. Schweitzer, D. Whitehead, G. Zweigle, and K. G. Ravikumar, "Synchrophasor-based power system protection and control applications," in *Protective Relay Engineers, 2010 63rd Annual Conference for*. IEEE, 2010, pp. 1–10.

[5] North American Electric Reliability Corporation, "Real-Time Application of Synchrophasors for Improving Reliability," *http://www.nerc.com/docs/oc/rapirtf/RAPIR%20final%20101710.pdf*, 2014, accessed: 2015-09-08.

[6] Americans for a Clean Energy Grid, "Synchrophasors," *http://cleanenergytransmission.org/wp-content/uploads/2014/08/Synchrophasors.pdf*, 2014, accessed: 2015-09-08.

[7] IEEE, "IEEE standard for synchrophasors for power syst." *IEEE Std C37.118-2005*, pp. 1–57, 2006.

[8] A. Daneels and W. Salter, "What is SCADA?" in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, 1999.

[9] J. Jiang, X. Zhao, S. Wallace, E. Cotilla-Sanchez, and R. Bass, "Mining pmu data streams to improve electric power system resilience," in *Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*. ACM, 2017, pp. 95–102.

[10] T. D. O'Rourke, "Critical infrastructure, interdependencies, and resilience," *BRIDGE-WASHINGTON-NATIONAL ACADEMY OF ENGINEERING-*, vol. 37, no. 1, p. 22, 2007.

[11] A. Giani, S. Sastry, K. H. Johansson, and H. Sandberg, "The VIKING Project: an Initiative on Resilient Control of Power Networks," in *The 2nd International Symposium on Resilient Control Systems (ISRCS 2009)*. IEEE, 2009, pp. 31–35.

[12] B. Miller and D. Rowe, "A Survey SCADA of and Critical Infrastructure Incidents," in *Proceedings of the 1st Annual conference on Research in Information Technology*. ACM, 2012, pp. 51–56.

[13] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May 2011.

[14] D. Kushner, "The real story of stuxnet," *IEEE Spectr.*, vol. 50, no. 3, pp. 48–53, March 2013.

[15] T. Chen and S. Abu-Nimeh, "Lessons from stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, April 2011.

[16] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke, "SCADA Security in the Light of Cyber-Warfare," *Computers & Security*, vol. 31, no. 4, pp. 418–436, 2012.

[17] K. Vu, M. Begovic, D. Novosel, and M. Saha, "Use of local measurements to estimate voltage-stability margin," *IEEE Trans. Power Syst.*, vol. 14, no. 3, pp. 1029–1035, Aug 1999.

[18] X. Jiang, J. Zhang, B. Harding, J. Makela, and A. Dominguez-Garcia, "Spoofing gps receiver clock offset of phasor measurement units," *IEEE Trans. Power Syst.*, vol. 28, no. 3, pp. 3253–3262, Aug 2013.

[19] Z. Zhang, S. Gong, H. Li, C. Pei, Q. Zeng, and M. Jin, "Time stamp attack on wide area monitoring system in smart grid," in *Comput. Res. Repository*, February 2011.

[20] D. Shepard, T. Humphreys, and A. Fansler, "Evaluation of the vulnerability of phasor measurement units to gps spoofing attacks," in *Int. Conf. Critical Infrastructure Protection*, Washington, DC, USA, 2012.

[21] D. Nguyen, R. Barella, S. A. Wallace, X. Zhao, and X. Liang, "Smart grid line event classification using supervised learning over pmu data streams," in *Green Computing Conference and Sustainable Computing Conference (IGSC), 2015 Sixth International*. IEEE, 2015, pp. 1–8.

[22] E. Klinginsmith, R. Barella, S. Wallace, and X. Zhao, "Unsupervised Clustering on PMU Data for Event Characterization on Smart Grid," in *Proceedings of the 5th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, 2016, pp. 1–8.

[23] S. Wallace, X. Zhao, D. Nguyen, and Kuei-Ti, "Big Data Analytics on Smart Grid: Mining PMU Data for Event and Anomaly Detection," in *Big Data: Principles and Paradigms*, R. Buyya, R. Calheiros, and A. Dastjerdi, Eds. Burlington, MA: Morgan Kaufmann, 2016, pp. 417–429.

[24] R. Mitchell, I. Chen *et al.*, "Effect of intrusion detection and response on reliability of cyber physical systems," *IEEE Trans. Rel.*, vol. 62, no. 1, pp. 199–210, 2013.

[25] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proc. ACM Symp. Appl. Comput.* ACM, 2004, pp. 420–424.

[26] S. Kher, V. Nutt, D. Dasgupta, H. Ali, and P. Mixon, "A detection model for anomalies in smart grid with sensor network," in *Future of Instrumentation Int. Workshop, 2012.* IEEE, 2012, pp. 1–4.

[27] J. Landford, R. Meier, R. Barella, S. Wallace, X. Zhao, E. Cotilla-Sanchez, and R. B. Bass, "Fast sequence component analysis for attack detection in smart grid," 2016.

[28] J. Magiera and R. Katulski, "Accuracy of differential phase delay estimation for gps spoofing detection," in *36th Int. Conf. Telecommun. and Signal Process.*, July 2013, pp. 695–699.

[29] P. Hines, S. Blumsack, E. C. Sanchez, and C. Barrows, "The Topological and Electrical Structure of Power Grids," in *Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS 2010).* IEEE, 2010, pp. 1–10.

[30] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[31] B. Yegnanarayana, *Artificial neural networks.* PHI Learning Pvt. Ltd., 2009.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[33] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intelligent Syst. and Technol.*, vol. 2, no. 3, p. 27, 2011.

[34] D. Britz, "Recurrent neural networks tutorial, part 1 introduction to rnns," 2015. [Online]. Available: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[35] A. Ng, "Sequence models." [Online]. Available: https://www.coursera.org/learn/nlp-sequence-models/home/welcome

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

[37] C. Olah, "Understanding lstm networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[38] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," in *The 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016)*, vol. 16, 2016, pp. 265–283.