DEEP LEARNING TECHNIQUES FOR DETECTION OF FALSE DATA INJECTION ATTACKS ON ELECTRIC POWER GRID

A Thesis

by

ARNAV KUNDU

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

MASTER OF SCIENCE

Chair of Committee,	Katherine Davis
Co-Chair of Committee,	Erchin Serpedin
Committee Members,	Le Xie
	Thomas Ioerger
Head of Department,	Miroslav M. Begovic

August 2019

Major Subject: Electrical Engineering

ABSTRACT

The electric power grid uses a set of measuring and switching devices for its operations and control. The data retrieved from the measuring instruments is assumed to be noisy, therefore a state estimator is used to estimate the correct values of state variables on which the system can take control actions. The modern electric power grid is dependent on communication networks for transferring these measurements, which are susceptible to intrusions from hackers. False data injection attacks (FDIA) are one of the most common attack strategies where an intruder tries to trick the underlying control system of the grid to cause disruptions without getting detected by native anomaly detection measures inbuilt in the state estimator. The native anomaly detection mechanism relies on threshold and residual based measure to flag a set of measurements as anomaly. Therefore, if the attack is devised in such a way that the intrusion can be performed without significantly affecting the residual error of state estimation it can go undetected. We propose a data augmented deep learning based solution to detect such attacks in real time. We propose methods of generating realistic random and targeted attack simulations on standard IEEE architectures and methods of detecting them using deep learning models. We propose recurrent neural network (RNN) based architectures to detect and locate FDIAs and devices compromised in real-time. For detection we propose a supervised and an unsupervised method. Similarly, for location we propose a method to find exact devices compromised which is less practical and then move on to a more feasible and practical solution in supervised and unsupervised conditions.

Being an intrusion detection system it is critical to detect all attacks which means false negatives should be penalized heavily, whereas false positives can be accommodated. Therefore, we use recall as our primary performance metric and precision recall curve to find an optimal threshold of probability score. In addition, we demonstrate how our approach is better than a residual error and other previous detection models. We also compare the performance of our models with increasing number of devices being compromised.

DEDICATION

To my parents, mentors and teachers.

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my advisor Dr. Katherine Davis for providing guidance and support throughout my graduate studies. I also wish to thank Dr. Le Xie who motivated to me to think about the ways neural networks can be used in power systems from his course "Data Science in Power Systems". I will also like to thank Dr. Erchin Serpedin and Dr. Thomas Ioerger for their service to my thesis committee.

A special thanks to my friends Vineet and Palash and my mentor at Nvidia Dr. Boris Ginsburg for many of the intense and deep discussions on deep learning and anomaly detection. Finally, I wish to thank all my family and friends in College Station and back in India for their constant support and encouragement.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professors Katherine Davis, Le Xie and Erchin Serpedin from the Department of Electrical and Computer Engineering and Professor Thomas Ioerger from the Department of Computer Science.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work was supported by National Science Foundation under Award #1446471.

NOMENCLATURE

FDIA	False Data Injection Attacks
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory
ROC	Receiver Operating Characteristics
AUC	Area Under Curve
PR curve	Precision Recall Curve

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

The electric power grid is a complex machinery involving multiple power generation sources, transmission lines and distribution substations which needs continuous monitoring for safe and reliable performance. This is done using a state estimator which helps to determine the best state of the system through a set of measurements and system models. The electric power grid has evolved immensely with time, and the modern smart grid uses an integration of information and communication technologies (ICT) and supervisory control and data acquisition (SCADA) for efficient remote monitoring and real-time control. These ICT relies on common infrastructure like internet, mainly because of its ease of access and distributed nature. Because of a common shared infrastructure, information flowing through this network can be extracted and modified by hackers. Such cases have been seen in the past like in Ukraine [1] an attacker opened circuit breakers to cause power outage by intruding into the SCADA system. Similarly, there have been instances of other attacks like the STUXNET [2], Pivnichna [3]. Since the electric power grid is a cyber-physical system, cyber-attacks can cause physical damage to the system. Common cybersecurity based approach, therefore, cannot help to secure the grid independently. Attacks can be of various forms like time synchronization attack [4], Denial of Service Attack (DOS)[5] which cause disruptions at the communication system while False Data Injection Attacks [6] (FDIA) cause disruptions at the physical system level.

FDIAs can be introduced in a transmission system to trick the state estimator into predicting wrong states without getting detected [7]. For intruders to launch such an attack, they need to know the complete configuration of the grid which appears to be highly unlikely, but it has been proven that such attacks are possible even with localized partial information [8]. The network can be protected from such attacks by two methods: protection of critical mea-

suring instruments and detection of attacks. Protection based methods try to find an optimal set of measuring devices which need to be secured and put on an encrypted channel [9]. But as the system starts getting larger, the number of devices to be secured starts growing which is infrastructurally infeasible [10]. Detection based methods try to find anomalies in the data received through the communication channel. Such methods depend on the real-time correlation between data points or the temporal structure of the data to classify a new set of measurements as anomalous. A significant drawback of this approach is that it does not adapt well to changing patterns in transmission behavior over time. FDIAs are challenging to detect using conventional residue based methods since they do not capture the spatial or temporal structure of the measurement data available. These methods were traditionally built to avoid bad data or severe measurement errors for DC state estimators where it is assumed that bad data will necessarily lead to high residual error. However, with the current case of FDIAs, we can ensure that bad data can be injected even by keeping low residual error. This is a classic contextual anomaly detection problem.

Deep learning has shown significant promises in solving complex tasks and has been used in pattern recognition problems like object detection, speech recognition, and anomaly detection. Deep learning uses a data-driven approach where a function approximator is trained using gradient descent over a given set of data points. The success of deep learning can be attributed to the ability of neural networks to learn complex functions and the availability of massive data-sets. Motivated by its application and success in the field of speech recognition, time-series prediction, and anomaly detection, we explore how recurrent neural networks can be applied to detect false data injection attacks in the electric power grid.

1.2 Literature Review

1.2.1 Defence Mechanisms

Since FDIAs are designed to bypass the bad data detection mechanism of the state estimator, a lot of innovative approaches have been studied in the past for capturing the spa-

tial and temporal correlations in the data to detect anomalous measurements. A traditional anomaly detection system can be designed as a prediction vs. real data problem [11]. In [12], the authors proposed to use independent data like forecast and historical patterns to detect anomalies. However, both of these methods depend on linear models; therefore, cannot be expected to operate correctly to capture nonlinearities induced in complex AC state estimation. They are also based on static thresholds which need to be carefully chosen and adapted over time. In [13], a tree pruning based approximation algorithm is used to detect anomalies in a graphical model. Inspired by various classical machine learning applications in cyber intrusion, sensor networks, and image processing, researchers have tried to apply nearest neighbor classifiers and other statistical classification techniques [14]. It is reasonable to assume that the attack vector is sparse. Therefore, sparse matrix reconstruction methods can be employed to identify devices compromised [15]. In [16], the authors assume that gradually changing state variables will typically lead to a low-rank measurement matrix Z_0 and the attack matrix (attack vectors over time) is sparse. Therefore, the problem translates to a matrix separation problem as:

$$\min_{\mathbf{Z_0},\mathbf{A}} \mathrm{Rank}(\mathbf{Z_0}) + \|\mathbf{A}\|_0, \quad s.t. \quad \mathbf{Z_a} = \mathbf{Z_0} + \mathbf{A} \tag{1.1}$$

which can be reformulated as a convex optimization problem as:

$$\min_{\mathbf{Z_0},\mathbf{A}} \|\mathbf{Z_0}\|_* + \lambda\|\mathbf{A}\|_1, \quad s.t. \quad \mathbf{Z_a} = \mathbf{Z_0} + \mathbf{A} \tag{1.2}$$

$\|\mathbf{Z_0}\|_*$ is the nuclear norm of Z_0, i.e., the sum of singular values of Z_0. This kind of optimization problem has been studied across the domains of compressive sensing and matrix completion and can be solved using off the shelf optimization algorithms. The problem with this approach is computational complexity because of its iterative nature [16]. The authors also propose a faster way using low-rank matrix factorization where low-rank matrix Z_0 is represented as a product of two matrices: U and V. However, such methods are not full proof

in defending against un-observable attacks [17]. Ozay et al. [18] formulated this anomaly detection problem as a classification task where attacked and non-attacked scenarios are separately labeled in the training phase. The authors experimented with methods like Sparse Logistic Regression, K-Nearest Neighbour (KNN) and Support Vector Machines (SVM) to classify a given set of measurements. Logistic Regression tries to find a hyperplane that separates the anomalous measurements from non-anomalous ones. This is only possible if the measurements are linearly separable. KNN uses a majority voting approach to classify features based on Euclidean distance. It is assumed that the Euclidean distance between anomalous measurements and non-anomalous measurements will be significant. SVM is a linear classifier that tries to find a hyperplane between two classes such that the distance between the two classes is maximum. It also allows the transformation of the actual feature space F into a high dimensional space S such that the features are linearly separable in S. A complex task in this approach is finding the dimensionality of S. When analyzed, the performance of these systems is impressive over the previous methods. However, these methods are slow for large networks and are not scalable. Besides, all these approaches are strictly supervised; therefore, the training data needed for these cases need to be extensive and cover all attack scenarios.

Generating all possible attack combinations for a large grid can be a challenging task. Therefore, creating an extensive data-set for supervised training of anomaly detection algorithms is not always feasible. Consequently, researchers have also proposed semi-supervised methods for detection of FDIA. In [19], a correlation based FDIA detection mechanism has been proposed. In this method, an operator needs to define a correlation sphere for various meters in the network. A single meter might lie in multiple correlation-spheres. At every iteration, correlations within a correlation-sphere are calculated, and if a considerable divergence is found, then an anomaly is flagged. This approach ensures that the spatio-temporal correlation between the measurements is preserved. This method is highly efficient in terms of run-time complexity but would need huge effort in designing the correlation spheres man-

ually. Also, this method will not allow adaptive change to network topology. Ozay et al. [17] proposed a semi-supervised SVM [20] where the contribution of labeled and unlabeled points are separately considered in the loss function. This kind of optimization problem setting helps to address unseen cases and generalizes better.

A power system characteristic based approach has been tested in [21] where a strategically selected set of devices are protected in order to make it impossible for an attacker to introduce an FDIA. This is because FDIAs are possible with complete certainty only if a certain number of devices are compromised [7]. The authors propose at finding the measurement devices to be protected using a brute force search and a basic measurement identification method. Another idea of detecting FDIAs is based on perturbation of line impedance and comparing measurements with expected measurements [22, 23]. This idea is based on the assumption that compromised measurement devices will not respond correctly to perturbations indicating presence of FDIAs.

Another approach for anomaly detection is based on unsupervised methods where the data-set is completely unlabeled for anomalies. Most of these methods rely on density based algorithms to find an approximate density cluster of non-anomalous data points. The points lying outside some margin of these density clusters are marked as anomalies [24]. For a dynamic system like the power grid, non-anomalous points can exist in different forms of density-clouds in an n-dimensional hyperspace which makes global density based detection infeasible. Therefore, the authors in [24] proposed to use local anomaly detection algorithms like Local Outlier Factor (LOF), Connectivity-based Outlier Factor (COF). Both of these algorithms use KNN to find regional clusters. It is assumed that the density of non-anomalous point clusters will be much higher than that of anomalous points, resulting in the identification of anomalies. However, since these methods are based on a non-parameterized algorithm, the run time complexity is high. In [24], the authors also proposed a Histogram-based Outlier Score (HBOS). This method is very similar to Naive Bayes algorithm [25] where the features are considered to be independent. For every feature, its histogram is con-

structed and then weighted by the inverse height of its bins. This gives a density estimate of all features. Since generating a histogram is faster than computing k neighbors for all data points, this algorithm is faster and performs reasonably for anomaly detection tasks. However, for application in electric power grid, the assumption of independence might hurt the purpose since most of the features are highly correlated. One class SVM [26] is another example of unsupervised anomaly detection where the classifier is trained to know what is usual or non-anomalous. This helps to form a boundary region for general data and anything lying outside those boundaries is flagged as an anomaly. But being a linear classifier, it is sometimes challenging to find an optimal kernel which can construct the correct margins for this method.

Artificial Neural Networks (ANN) have shown significant performance in representing complex functions [27]. With the advent of Graphics Processing Units (GPUs) and availability of massive data-sets, neural networks have helped to solve complex problems in the fields of object recognition [28], speech recognition [29] and anomaly detection [30]. Especially in anomaly detection, deep neural networks have been applied in many applications like fraud detection, sensor network anomaly detection, video surveillance, log anomaly detection and Internet of Things (IoT). Deep neural networks have been used in supervised [31], semi-supervised [32] and unsupervised setting [33] in the past for anomaly detection. Specifically for anomaly detection in spatially and temporally correlated data, direct supervision using classification networks and unsupervised methods using auto-encoders have shown impressive results in the past. Applications of the same has been shown in detection of electricity theft [34, 35]. Generative methods like Generative Adversarial Networks (GANs) [36] and Variational Auto-Encoders (VAEs) [37] have also been explored. Both of them are unsupervised methods where neural networks are trained to learn the latent distribution of non-anomalous data. GAN uses a discriminator to judge whether a new set of data points are different from the old set of data on which it was trained on. VAE uses the reconstruction error of the test set of points to find anomalies. Since we are dealing with a

contextual anomaly detection problem, we need to design a type of network that captures temporal correlations well. The reason behind such an assumption is that the power flowing through the grid follows a temporal structure. As mentioned earlier, we can use RNNs to model temporal structures; in the following section, we will explain how RNNs are trained and how improvements to classical RNN-cells can be achieved.

1.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have been used in multiple time-series models involving neural networks like stock price predictions, language models, etc. RNNs have a memory component which can store previous inputs and outputs to predict the next state. An RNN tries to map a sequence of inputs $x \in x_0, x_1, ..., x_t$ to a sequence of outputs $y \in y_0, y_1, ..., y_T$ where $T \geq 1$. Being a dynamic system, RNN-cells use a feedback mechanism to encode the temporal structure of the sequence so that subsequent outputs depend on current inputs. This is done using a hidden state at every time-step which is considered while computing the output at the next state. The following equations explain how outputs and hidden states are computed at every time-step.

$$h_t = \tanh(W_h \times h_{t-1} + W_i \times x_t) \tag{1.3}$$

$$o_t = \texttt{sigmoid}(W_o h_t) \tag{1.4}$$

Neural networks are trained using back-propagation algorithm. Back-propagation uses the chain rule to propagate gradients of prediction error (loss) to adjust all weights of the network [38]. For an RNN-cell, the outputs depend on its inputs and the previous hidden state. Therefore, an RNN-layer is expanded into continuous RNN-cells connected end to end forming a chain of fully connected layers as shown in Figure 1.1. When back-propagation is applied on this unrolled chain, it is referred to as Back Propagation Through Time (BPPT) [38, 39].

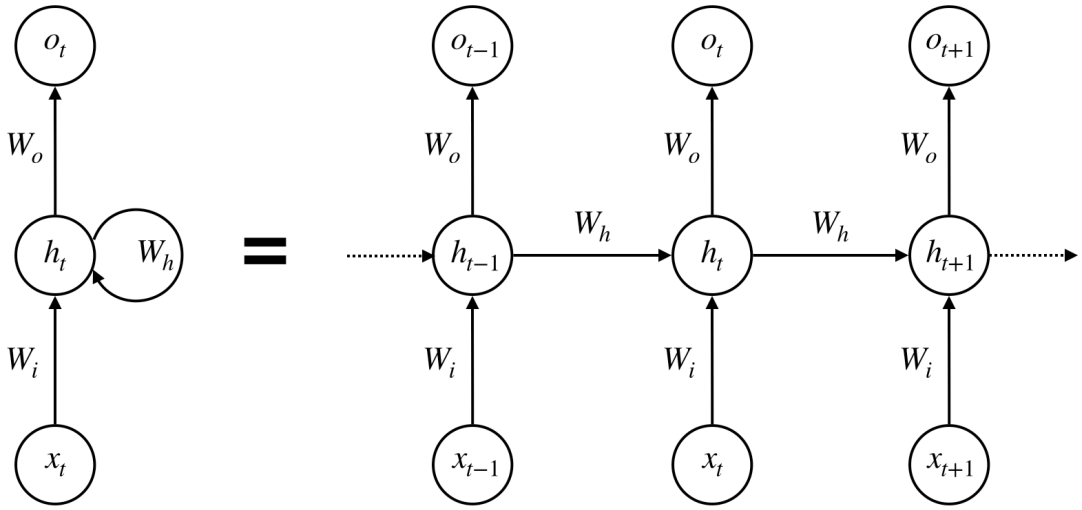

Figure 1.1: Unrolled RNN-cell

Let us assume the loss for a given input x and target output y is given as:

$$\mathcal{L}(x, y) = -\sum_t y_t \log o_t \tag{1.5}$$

where o_t is a function of x. In order to perform gradient descent we calculate the gradient of the loss with respect to W_h, W_o and W_i. For W_h we find the gradients as:

$$\begin{aligned}
\frac{\delta \mathcal{L}}{\delta W_h} &= \sum_t \frac{\delta \mathcal{L}}{\delta o_t} \times \frac{\delta o_t}{\delta h_t} \times \frac{\delta h_t}{\delta W_h} \\
&= \sum_t \frac{y_t}{o_t} \times \frac{\delta o_t}{\delta h_t} \times \frac{\delta h_t}{\delta h_{t-1}} \times \frac{\delta h_{t-1}}{\delta W_h} \\
&= \sum_t \frac{y_t}{o_t} \times \frac{\delta o_t}{\delta h_t} \times \prod_k \frac{\delta h_k}{\delta h_{k-1}} \times \frac{\delta h_0}{\delta W_h}
\end{aligned} \tag{1.6}$$

In a similar manner, gradients are calculated for W_o and W_i. It can be seen from the equation 1.6 that calculation of gradient involves calculating a product of gradients of all hidden states with respect to their previous hidden states. This causes a problem of vanishing gradients [40] in simple RNN based networks which nullifies the effect of initial inputs on the final output. Alternatives like Gated Recurrent Units (GRU) [41] and Long Short Term Memory

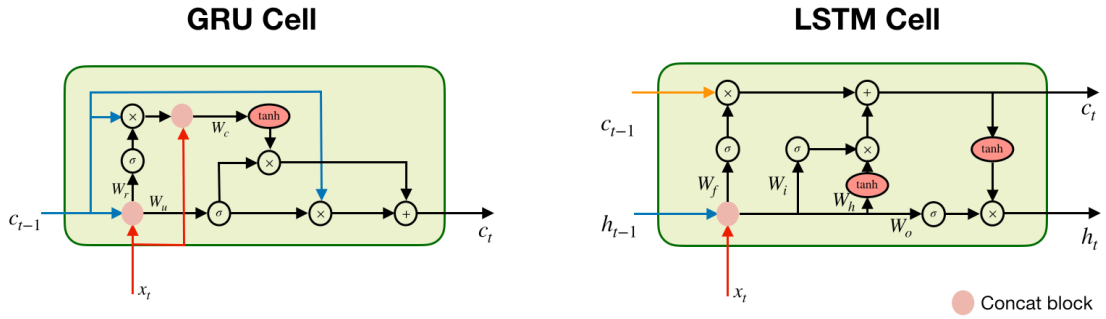

Figure 1.2: GRU and LSTM Cells

(LSTM) [42] have been proposed to get rid of this problem. GRUs introduce the concept of 'context,'(c_t) which is composed of a mixture of past context and a function of present input as shown in equation 1.7.

$$\begin{aligned} G_u &= \sigma(W_u[c_{t-1}, x_t]) \\ G_r &= \sigma(W_r[c_{t-1}, x_t]) \\ \tilde{c}_t &= \tanh(W_c[G_r * c_{t-1}, x_t]) \\ c_t &= G_u * \tilde{c}_t + (1 - G_u) * c_{t-1} \end{aligned} \tag{1.7}$$

If G_u is zero $\forall\ t$, a direct path of information flow can be drawn from the first input to the last context. Since G_u is controlled by trainable weights W_u, we can expect the network to learn the optimal mixture for capturing long term dependencies. GRUs have become lately popular primarily due to the ease in training them and lesser memory consumption over LSTM. This is because LSTMs maintain different update and forget gates. Update gates are used to tune $\tilde{c}_t$ and forget gates are used to calibrate c_{t-1}. Basically, in LSTMs contribution of parameters at t and $t-1$ are independent. The block diagrams of GRU and LSTM cells are shown in Figure 1.2

GRUs and LSTMs are the fundamental blocks of our temporal models. Another important concept used in semi-supervised anomaly detection neural network is 'auto-encoders.'

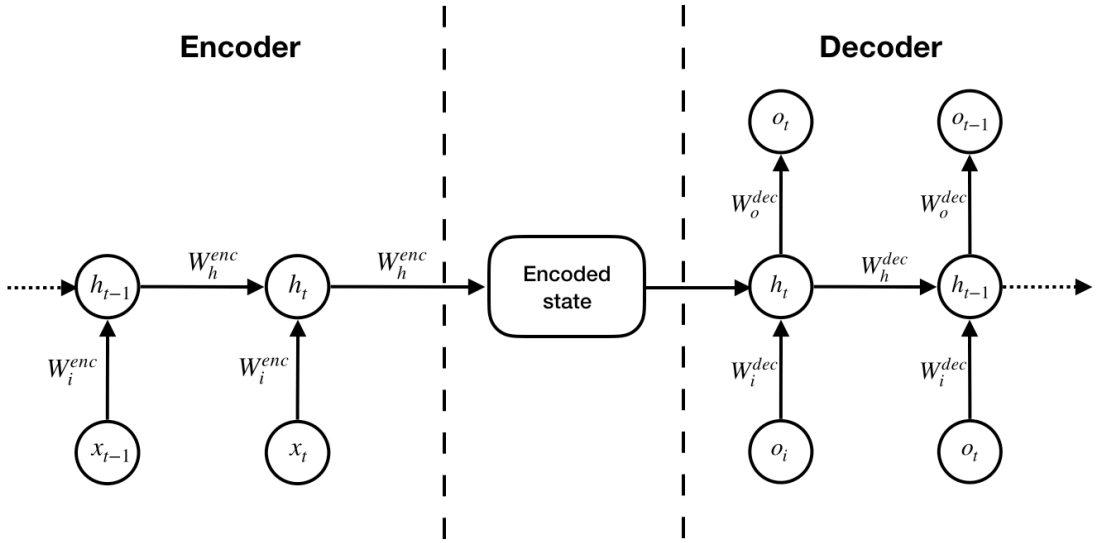

Figure 1.3: Sequence to Sequence Auto-Encoder

We will take a look into auto-encoders and ways to train them in the following section.

1.2.3 Auto-Encoder and Attention

In a semi-supervised setting, a neural network is trained on non-anomalous data to compress input measurements into an encoded state. The encoded state is used to predict the input measurements using a decoder. This mechanism is called an auto-encoder [43]. Since we are dealing with temporal correlations, it will be correct to use a sequence to sequence model as auto-encoder. In a classical sequence to sequence auto-encoder model [44], the encoder encodes the entire sequence in its hidden state at the last time step. This hidden state is then fed into a decoder to predict the input sequence. The architecture is illustrated in Figure 1.3. The decoder uses the current hidden state (h_t) and output (o_t) to predict the previous hidden state (h_{t-1})and output(o_{t-1}). In RNN (GRU, LSTM) it is expected that the hidden state or context at time t encodes the entire time series. But in practice, this is not always true. In a power system, it can be ideal to assume that there exists a monotonic temporal correlation, i.e., the past influences the future. Therefore, the model should be able to encode past measurements ($x_0, x_1, ...x_{t-1}$) with equal importance as present measurements

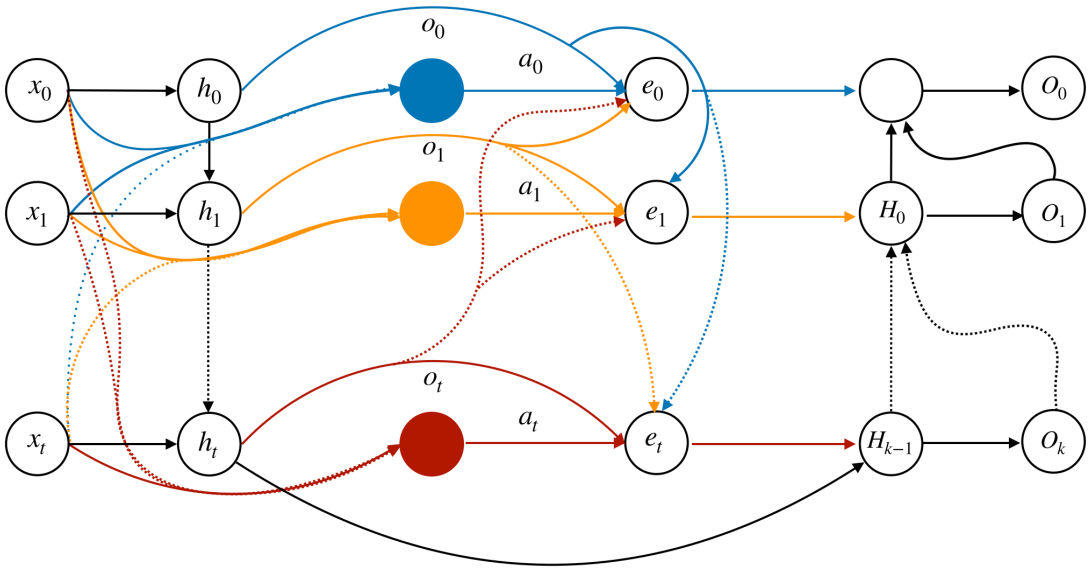

Figure 1.4: Implementation of Attention in Sequence to Sequence Model

(x_t). This is achieved with the help of attention [45]. Attention mechanism was developed for machine translation where a one to one relationship between inputs and outputs do not exist. n words in a source language can correspond to p words in a target language. Additionally, the arrangement of words might also change. Attention allows us to draw a correlation between input and output sequences by weighing the contribution of every input sequence element to every output sequence element. Figure 1.4 illustrates implementation of attention in a sequence to sequence model. This version of attention is called global attention. For this model, output at time-step n depends on both the past and the future inputs of the sequence i.e. $O_n = f(x_i, h_i)$ where $i \in (0, t)$, where t is the input sequence length. But for a real time system, inputs can only have past and present values. Therefore, monotonic attention is used in such cases. Monotonic attention [46] represents a causal system where $O_n = f(x_i, h_i)$ where $i \in (0, n)$. For a power system analysis, monotonic attention can help to find the importance of past measurements in predicting the present measurements.

1.3 Anomaly Detection Metrics

Generally for binary classification tasks like anomaly detection, common metrics for judging the performance of a model are Accuracy, Precision, Recall, F1-Score, ROC-AUC and Precision-Recall curve [47]. Since the frequency of anomalies is much lesser than that of non-anomalous conditions, therefore, there is an inherent class imbalance in the data-set. Hence, metrics like accuracy will not be a good evaluation parameter, while precision and recall can be good parameters. Precision captures the correctness of all positively classified points whereas recall captures the ratio of the number of positive classification and total positive points present. Being an anomaly detection system, we need to have very high recall and can trade off with precision. F1-score is calculated by the harmonic mean of precision and recall. ROC-AUC and Precision-Recall curve both help to find an optimal threshold for classification.

1.4 Roadmap

Moving forward we will discuss about FDIA generation and implementation algorithms in Chapter 2. Then will be discussing our proposed methodologies of detection and location of attacks along with the pros and cons of each of them in Chapter 3. We will discuss our experimental setup and observations in Chapter 4 followed by conclusion and possible extensions of our methods in Chapter 5

2. GENERATION OF ATTACKS

The basic concept behind FDIAs is very simple [7], i.e., to generate an attack vector a such that:

$$z + a = H(x + c) + \epsilon \tag{2.1}$$

where c is the change in states induced due to the attack vector, $z \in \mathbb{R}^{n\times 1}$ is the actual measurement vector from n devices, x is the state vector corresponding to correct measurements and H is the system defination matrix. We have experimented on the generation of two types of FDIAs: Random and Targeted Attacks.

2.1 Attack Generation Algorithms

2.1.1 Random Attacks

One of the simplest attacks is random attack where an attacker with access to a fixed set of compromised measuring devices tries to bias random state variables from a random probability distribution. The following derivation shows why such an attack is possible [7].

$$\begin{aligned} \hat{x}_a &= (H^T \Sigma H)^{-1} H^T \Sigma z_a \\ &= (H^T \Sigma H)^{-1} H^T \Sigma (z + a) \\ &= \hat{x} + (H^T \Sigma H)^{-1} H^T \Sigma a \end{aligned} \tag{2.2}$$

Here, $\hat{x}$ represents the state vector under normal conditions and $\hat{x}_a$ represents the state vector under attack, H is the system definition matrix, Σ corresponds to the allowed margin of error for every measurement. $z_a = z + a$ represents the attacked measurements values received by

the state estimator, where z is the actual measurement and a is the attack vector.

$$\begin{aligned} ||z_a - H\hat{x}_a|| &= ||z + a - H(\hat{x} + (H^T\Sigma H)^{-1}H^T\Sigma a)|| \\ &= ||z - H\hat{x} + a - H(H^T\Sigma H)^{-1}H^T\Sigma a|| \\ &= ||z - H\hat{x} + a - H(H^T\Sigma H)^{-1}H^T\Sigma Hc|| \\ &= ||z - H\hat{x} + Hc - Hc|| \\ &= ||z - H\hat{x}|| \end{aligned} \tag{2.3}$$

Such an attack is possible straight away if the attacker has access to all the measuring instruments. However, it is tough to get hold of all the measuring devices in a network. As a result, we cannot choose any random attack vector. Let $I_{meter} = \{i_1, ..., i_k\}$ be the set of indices of the k meters the attacker has access to. Therefore, $a = (a_1, ..., a_m)^T$ with $a_i = 0$ for $i \notin I_{meter}$ [7]. In order to find one such attack vector we define a projection matrix

$$P = H(H^T\Sigma H)^{-1}H^T\Sigma \tag{2.4}$$

From the previous equation we can derive the following relations:

$$\begin{aligned} a - H(H^T\Sigma H)^{-1}H^T\Sigma a &= 0 \\ Pa &= Ia \\ (P - I)a &= 0 \\ Ba &= 0 \end{aligned} \tag{2.5}$$

Therefore, an attacker needs to find a non zero attack vector a such that $Ba = 0$ and $a_i = 0$ for $i \notin I_{meter}$. Let us represent

$$\begin{aligned} a &= (0, 0, ..., a_1, 0, 0..., a_2, 0, ..., a_3, ...a_k...)^T \\ B &= (..., b_{i1}...b_{i2}, ...b_{ik}...) \end{aligned}$$

where a_i is the attack corresponding to the ith meter for $i \in I_{meter}$ and b_i is the column vector in B corresponding to the index of a_i in a. Therefore, we define $B' = (b_{i1}, b_{i2}, ..., b_{ik})$ and $a' = (a_1, a_2, ..., a_k)$ such that $Ba = 0$. If the rank of B is less than k, then there can be infinitely many solutions to $Ba = 0$. According to Meyer [48] a' can be determined as:

$$a' = (I - B'^{-1}B')d \tag{2.6}$$

where d is some random non-zero vector. If rank of B' is greater than or equal to k, then there is only one unique solution to $Ba = 0$ i.e., $a = 0$. Therefore, it can also be logically inferred that the probability of generating a random attack increases if we have access to more meters [7].

2.1.2 Targeted Attacks

In a targeted attack, the attacker wants to control particular state variables. We can represent one such attack mathematically as follows:

$$I_{states} = \{i_1, ..., i_k\} \text{ where } k < n \tag{2.7}$$

Which denotes the state variables to be attacked. The objective of the attacker is to inject an attack state vector c such that $\hat{x}_{bad} = \hat{x} + c$ where $c = (...c_1, ..., c_k...)^T$. We consider two cases of attacks over here: a constrained case and an unconstrained case. A constrained case is one where we assume that the injected attack does not affect any other state apart from the targets. In the unconstrained case, it is assumed that the attacker does not care about the impact of his attack on other state variables apart from his targets.

2.1.2.1 Constrained Attacks

In constrained attacks, the attacker already knows the change needed to be introduced in the state variables (c) and the other state variables aren't supposed to be affected in any manner. Therefore, we define $c = (0, ..., c_1, 0, ..., c_k, 0)$ where c_i denotes the state variables,

we want to change. The attack vector can be directly determined according to the equation $a = Hc$ [49]. The generated attack vector is then compared with the measuring devices that are compromised, if all the non-zero entries in a are accessible then an attack is possible.

2.1.2.2 Unconstrained Attacks

In an un-constrained attack an attacker aims to generate an attack vector a [49] such that

1. $a = Hc$
2. $a_i = 0 \forall i \notin I_{meter}$
3. $c = (...c_1, ..., c_k...)^T$

Such an attack vector is derived as follows:-

$$\begin{aligned} a &= Hc \\ a &= \sum_{i \notin I_{states}} h_i c_i + \sum_{j \in I_{states}} h_j c_j \\ a &= H_s c_s + b \\ a - b &= H_s c_s \\ H_s (H_s^T H_s)^{-1} H_s^T (a - b) &= H_s (H_s^T H_s)^{-1} H_s^T H_s c_s \\ H_s (H_s^T H_s)^{-1} H_s^T (a - b) &= a - b \\ (H_s (H_s^T H_s)^{-1} H_s^T - I) a &= (H_s (H_s^T H_s)^{-1} H_s^T - I) b \\ B_s a &= B_s b \end{aligned} \tag{2.8}$$

where,

$$b = \text{impact of attack states}$$

$$B_s = H_s (H_s^T H_s)^{-1} H_s^T - I$$

$$H_s = H_{[:,j]} \forall j \notin I_{states}$$

Next a subset of columns is derived from B for the indices for which $i \in I_{meter}$ as B like in the random attacks scenario. Therefore, as derived earlier $B'_s a' = B_s b$ [49]. The attacker can solve for a and finally derive an attack vector for the targeted attack. However, it is not guaranteed that an attack vector would be possible from any random set of compromised meters.

2.2 Attack Generation and Storage

Using the two types of attack formation algorithms mentioned above, we generate a database of attacked and non-attacked scenarios. The simulation uses real-world power consumption data to generate measurements and the intrusion state of these devices at each time step. The attack data is stored along with the state variables under attack and the devices compromised. This is treated as the training data for our deep learning models.

In an ideal power system, attacks are rare. Also, it is highly unlikely that in all scenarios where attacks are possible, the attacker has access to all measurement units. Once an attack is introduced, it can stay for a variable amount of time. Therefore, while generating simulations, we consider these factors in choosing the frequency, duration, and location of these attacks. We have created cases to select a fraction of random devices parameterized by k from n measuring devices ($k \in 0.1, 0.2, 0.3, 0.4, 0.5$). Similarly, we have assigned a probability (p) of the grid being under attack where $p \in [0, 0.2)$ and we have kept each attack live for a random number of samples ($t \in$[5,10]). This provides a huge range of possible combinations to store in our database of attacks. The algorithm used to generate these attack data-sets is described in Algorithm 1. For targeted attacks, it might not be possible to generate an attack vector for random combinations of compromised meters. Hence, we try to find the change in the state vector, i.e., c that can be induced by having access to the selected measuring devices. We use the same vector as targets.

Algorithm 1 Generation of Attacks

```
1:  procedure GENERATEATTACK(attackType, measurements, devices)
2:      for i in range(sizeOf(measurements)) do
3:          options ← [0.1, 0.2, 0.3, 0.4, 0.5]
4:          k ← choice(options, 1)
5:          p ← random(0, 1)
6:          t ← randomInt(5, 11)
7:          hacked ← choice(devices, int(k × sizeOf(devices))
8:          if p < 0.2 then
9:              for j in range(t) do
10:                 z ← measurements[i]
11:                 if attackType ≠ "random" then
12:                     a, target ← getTargetAttack(hacked, t)
13:                     zNew ← z + a
14:                     saveRecords(zNew, hacked, target)
15:                 else
16:                     a ← getRandomAttack(hacked, t)
17:                     zNew ← z + a
18:                     saveRecords(zNew, hacked)
19:                 end if
20:                 j+ = 1, i+ = 1
21:             end for
22:         else
23:             z ← measurements[i]
24:             saveRecords(z)
25:         end if
26:     end for
27: end procedure
```

The critical contribution of this thesis is the detection and location of intrusions for FDIAs. Detection corresponds to informing a system operator, or the state estimator about an FDIA and location aims at identifying the devices which have been compromised. We propose a few approaches in the detection and location of FDIAs.

3.1 Detection

We propose a few approaches to finding the cases when an attack is introduced in the system. These approaches are based on sequence to sequence models. As discussed earlier, RNNs help to encode sequences in a condensed latent space. This property can be exploited to detect intrusions as well. Intrusion detection for a spatially and temporally correlated system like the electric power grid using RNNs can be done using a completely supervised approach or an unsupervised approach. We have implemented a variant of both the methods and compared their performance and feasibility.

3.1.1 Fully Supervised Global Detector

As mentioned earlier, the detection of FDIAs can be modeled as an anomaly detection problem. This is because attacks are generally sparse vectors [16] added to the actual measurements violating the temporal structure of the data. Therefore, a sequence model should be able to detect such anomalous patterns. We use a GRU based many to one sequence to

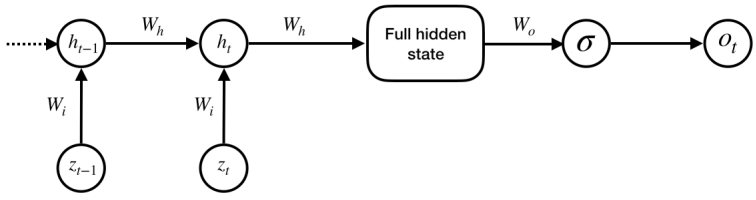

Figure 3.1: Architecture of Fully Supervised Global Detector Network

sequence model, which takes in measurements over a window of length k from time-steps $[t-k, t]$ to predict whether the measurements at time-step t is anomalous or not. The system architecture is shown in Figure 3.1. We believe the temporal structure of the dynamic power system is captured by the weights W_h. W_i captures the effect of individual measurements on the latent space. For a power system the latent space can be imagined as a compressed version of the state vector and W_i captures the impact of every measurement on the latent space h. The final hidden state is passed through a linear layer W_o which compresses hidden state vector $h_t \in \mathbb{R}^{h \times 1}$ to $\mathbb{R}^{1 \times 1}$. Finally, we use a sigmoid function to compute a probability score that corresponds to an attack being present at time-step t.

This network is trained on anomalous ($o_t = 1$) and non anomalous data ($o_t = 0$) generated by Algorithm 1. We have assumed that the probability of the grid being under attack is far less than that of normal operating conditions. For 0-1 prediction tasks, binary cross entropy loss is the best maximum likelihood estimator [50]. Since neural networks are trained on empirical loss minimization, the predictor might not even predict anomalies at all because of their infrequent nature. We avoid such a condition from happening by weighing the loss function so that miss in predicting anomalies (false negatives) are penalized more than miss in predicting normal operating conditions (false positives). The loss function for our classification model is described in equation 3.1, where a is the penalty multiplier for false negatives. Since neural networks are trained using mini-batch stochastic gradient descent, a possible option is to sample anomalous states more often while constructing mini-batches to ensure an equal distribution of anomalies for the network to learn. But this will skew the actual distribution of anomalous states, therefore, inducing an incorrect bias in the model.

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{N} a \times y \log(\hat{y}) + (1 - y) log(1 - \hat{y}) \tag{3.1}$$

3.1.1.1 Data Preparation and Training

Since this is a sequence to sequence model, we need to prepare mini-batches of time-series and the corresponding state of the system for the last time-step of the sequence. We use a rolling window of fixed length to create the training pairs. The inputs are measurements over a time window and the output corresponds to the state of the system at the last time-step of the window. The model is trained using mini-batch gradient descent by an ADAM [51] optimizer. To ensure that the model does not overfit on the training data we use dropouts [52]. Dropouts randomly drops neural connections between layers making the model robust to noise. It also prevents the neural network to get over reliant on specifics neurons, therefore helping in regularization.

3.1.2 Unsupervised Global Detector

Generation of extensive training data covering all attack scenarios is tricky. With increasing grid sizes the range of possible attacks that can be introduced by varying the combination of compromised devices is very large. It is practically infeasible to cover all attack scenarios. Unsupervised methods, on the other hand, do not need labeled training data. Therefore, there is no need for generating massive training datasets.

We propose an auto-encoder based sequence to sequence model. As mentioned earlier, an auto-encoder comprises of two parts: the encoder and the decoder. The encoder computes a latent representation of the entire sequence in a condensed form. This is decoded by the decoder to restore the original input sequence. The auto-encoder is expected to learn a dimensionality reduction function for the encoder and a restoration function for the decoder. The idea behind using this kind of a structure is that the model can learn inter-dependencies between measurements in the encoder to find an optimal hidden state and that can be restored using decoder parameters. Since we are dealing with an over-determined system, therefore, we can assume that restoration of measurements from a compressed hidden state is tractable. But as mentioned earlier, a vanilla sequence to sequence auto-encoder might not be able to

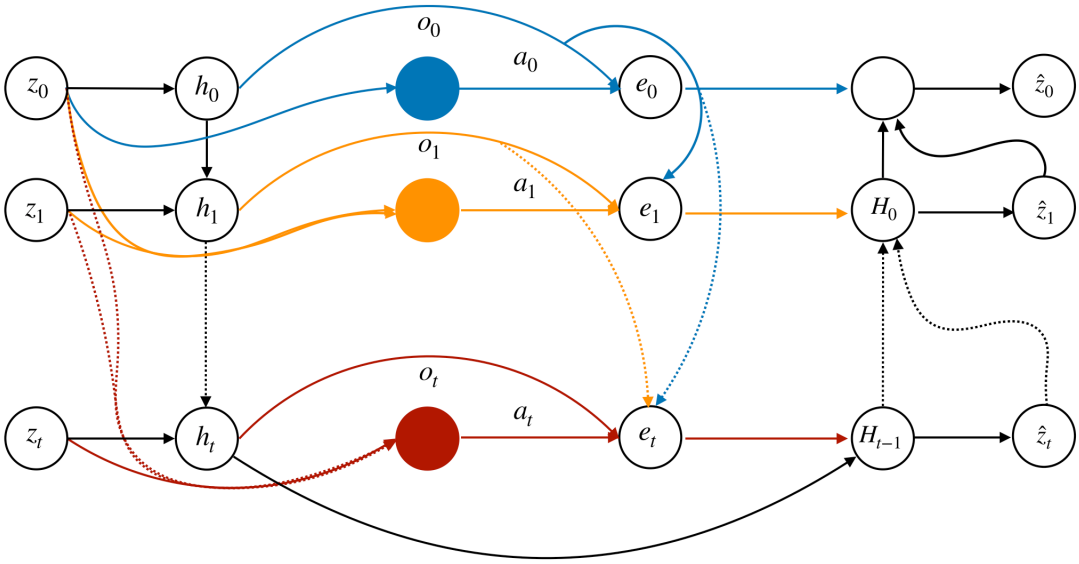

Figure 3.2: Architecture Explaining Sequence to Sequence Auto-Encoder with Monotonic Attention

model long term dependencies. Therefore, we use monotonic attention in our sequence to sequence auto-encoder. Monotonic attention helps the decoder to learn the importance of every past state to a given output state. This model is trained on non-anomalous data, to minimize the mean reconstruction loss. The optimization function is shown in equation 3.2, where D represents the decoder function and E denotes the encoder function.

$$\begin{aligned} &\min_{\theta_D,\theta_E} \frac{1}{t}\sum_{t}(z_t - \hat{z}_t)^2 \\ &\text{s.t. } \hat{z}_t = D(\theta_D, E(\theta_E, z_t, z_{t-1}, ..., z_0)) \end{aligned} \tag{3.2}$$

We train this network until reconstruction error is very close to zero. This ensures that our model is fully aware of what normal conditions look like and how measurements are correlated to one another. We hypothesize that for anomalous data, i.e., when an attack vector is added to actual measurements, the reconstruction error will be high. This is because FDIA vectors are normally sparse; hence the encoded representations should not be significantly

affected. This would make the decoded measurements $\hat{z}_t$ appear similar to non-attacked measurements z_t instead of $z_t + a$. It can be seen from Figure 3.2 that the output $\hat{z}_t$ is a function of e_t and h_t. e_t is a function of attention vector a_t and $(h_0, h_1, ..., h_t)$. Therefore, it might appear that the attention vector might force the contribution of the encoder to zero and completely pass the inputs at each time instance with an identity mapping to the output. In other words, it might appear that the attention weights $W_{ai}^t = 0, \forall i \neq t$ and $W_{ai}^t = 1$, if $i = t$; while the encoder hidden vector weight $W_h = 0$. However, while training we decode the entire sequence with the help of a GRU decoder and combining previous outputs along with present inputs. We can ensure that the contribution of the encoder is not nullified i.e., $W_h \neq 0$.

3.1.2.1 Data Preparation, Training, and Inference

Since this is a unsupervised method, we do not need labeled attacked and non-attacked cases. Instead, we train the auto-encoder only on non-attacked data. Since it is a sequence to sequence auto-encoder while training we use the same sequences of a fixed length as input and target as illustrated in Figure 3.2. To prepare the training sequences, we use a rolling window of a fixed sequence length for all the measurements. The RNN-decoder uses previously decoded outputs at every time-step for computing the output for the next time-step. We use forced training by feeding actual targets at each time-step instead of predictions. Practically this helps the network to converge faster and has lower reconstruction error.

While inference, we are not concerned about the outputs at time-step $(0, t-1]$. We only care about the outputs at the last time-step t. We find the reconstruction error by calculating the mean squared error between the input measurements at time-step t and the corresponding output of the auto-encoder. If the reconstruction error is above a given threshold, we flag it as an anomaly. We will explain the idea behind estimating the threshold later. Another significant difference from the training step is that during inference, we use the predictions from the auto-encoder as true measurements to replace anomalous measurements. This helps to keep the sanctity of the detection mechanism for the subsequent time-steps.

One approach to decide the threshold for classification of anomalies can be decided by the mean (μ) and standard deviation (σ) of reconstruction error under ideal operating conditions. We can use statistical confidence based approach where we can assume the reconstruction error (L) to follow a normal distribution and flag values outside some multiples (k) of standard deviation from the mean as anomalies as shown in Equation 3.3. k can be varied to plot a precision-recall curve, and an optimal threshold can be determined.

$$\begin{aligned} |L - \mu| &> k \times \sigma \rightarrow 1 \\ |L - \mu| &<= k \times \sigma \rightarrow 0 \end{aligned} \tag{3.3}$$

Another approach might be to train another function approximator in a supervised setting on reconstruction error using the training data used in the fully supervised case. Since we want to study the results for a completely unsupervised setting, therefore, moving forward, we will not be considering this approach.

3.2 Location

So far we have seen how a global detection mechanism using auto-encoders and strictly supervised methods are realized. In this thesis, we take a step forward to locate the points of intrusion and find compromised devices. This will not just help in protecting the power grid from FDIAs but also help to ensure proper functioning without disruptions by ignoring the compromised measurements for state estimation. Also, it will help to identify possible points of intrusion, therefore, help operators to take protective measures to secure most vulnerable nodes.

3.2.1 Unsupervised Location

We have already mentioned that the predictions of the auto-encoder should be very close to usual grid operating conditions even if in the last time-step an intrusion exists. Therefore, the reconstruction error can be a good indication of the actual attack vector injected. But since the auto-encoder has some function approximation error, we cannot expect the recon-

struction error to be an accurate estimate of attack vector. Instead, we can use a threshold based detection method to classify devices as attacked or unattacked based on their respective reconstruction errors. All the measurements are represented in their per unit values. Therefore, based on grid conditions, the variance of each measurement would be unique, which makes it challenging to have a global threshold for classification. Thus, we use the mean and standard deviation of the approximation error for each measurement device separately under ideal operating conditions. Based on these mean and standard deviations, we decide the classification thresholds of classifying each device as compromised. Similar to the previous case an optimal threshold can be found using equation 3.3 and precision-recall curve. Also, many other methods can be considered to classify anomalies based on reconstruction error, but we moved forward with the simplest architecture and studied its performance.

Using this approach, we can detect intrusions and locate devices compromised at the same time using the same model. Since it is trained on ideal grid operation data, we do not need an extensive set of simulated attack data to train it. Also, it can help the state estimator with approximately correct measurements in case of long intrusions, thereby facilitate normal grid operations. The only problem we believe might occur is in choosing the correct hyper-parameters like the hidden layer size, sequence length.

3.2.2 Supervised Location with Localized Measurements

All the approaches discussed so far work on a global idea where all measurements are considered together for predicting the state of the system. We also assume the power grid network configuration to be fixed. This restrains our model from being flexible to network architectures. If the network architecture changes, we need new training data to train our model to adapt to the latest measurement values. Since we are discussing a more practical solution, locating specific devices sounds a critical step but it might not be the most practical one. Instead, if we aim at identifying anomalies strictly on the measurement devices connected to a given bus independently, it can be a reasonable estimate of the location intrusions. Therefore, each bus can maintain its copy of the model trained for a particular configuration

on the measurement devices connected to it. If all the devices connected to a given bus are compromised, it will be highly unlikely of the model to predict an anomaly going just by the spatial correlations. Therefore, we need to consider temporal patterns in the data as well for developing our model. In case only a few devices connected to a bus are compromised, the spatial characteristics can also play a role in the identification of anomalies. But overall, the model needs to rely more on the temporal structure.

We use a similar approach of using an RNN based detection mechanism as discussed in the fully supervised global detection method above. The only difference here is the input vector and the size of the dimension of the hidden state. Since we do not depend a lot on the spatial correlations, we do not compress the input measurements into a latent vector to encode the spatial relationships between them as in the previous case.

3.2.3 Unsupervised Location with Localized Measurements

Similar to training local supervised models for detecting intrusions at every bus we can also adopt an unsupervised approach. In this method, we train the attention based auto-encoder only on the measurement devices connected to a particular bus. This allows every bus to maintain its own model, therefore, solving the problem with scalability. As done in the case of detection, we use the reconstruction error of the auto-encoder to classify intrusions on every bus independently.

But, this approach might face some problems when most of the devices connected to a given bus are compromised. This is because the reconstruction error in that case will not be high. The reason behind it is that for unsupervised global detection using auto-encoder we assumed that the attack vector is sparse, therefore, the contribution of attack will not skew the outputs from the auto-encoder and they will be close to normal operating conditions. However, in the localized case the input measurements can all be under attack, therefore, the attack vector does not stay sparse any more. Such a case might occur when most of the devices on a given bus are compromised.

4. EXPERIMENTS AND OBSERVATIONS

4.1 Data Generation and Test Cases

For a realistic study of grid conditions under attack and non-attacked scenarios, we gathered real-world power consumption data. This data is used on an IEEE 14-bus case (Figure 4.1). to compute AC power flow and measurements of all 39 devices available from standard MATPOWER simulation. We gather all this data as our baseline ideal grid operating conditions. An example for Bus 1 is shown in Figure 4.2. In the next step, we generate attacks on these ideal operating conditions using the algorithm discussed in Chapter 2. For targeted attacks, it was observed that under many combinations of compromised devices, the attack was not feasible. Therefore, we used random attacks as a case of targeted attacks where the targets were the state variables affected by a given random attack vector. Thus our final problem translates to detecting and locating any random attack. Also, we can claim that all attacks are a subset of random attacks. Therefore, if we can detect random attacks correctly, we should be able to identify other kinds of attacks as well. As mentioned earlier, we use multiple levels of intrusion starting from 10% to 50% of devices getting compromised. For each of these intrusion levels, we generate $250,000$ points of attack data to extensively cover all possible scenarios.

4.2 Detection

For the fully supervised model, we use 500000 (100000 from each level of intrusion) data points as the training set and 125000 (25000 from each level of intrusion) points as the validation set. After every epoch on the training set, the validation loss is computed on the validation set. If the validation loss is lesser than previously calculated validation losses, then the model is saved. The model is trained for 200 epochs. Then the inference is run individually on the remaining 125000 points from each level of intrusion data. As discussed

[1] source: https://electricgrids.engr.tamu.edu/electric-grid-test-cases/ieee-14-bus-system

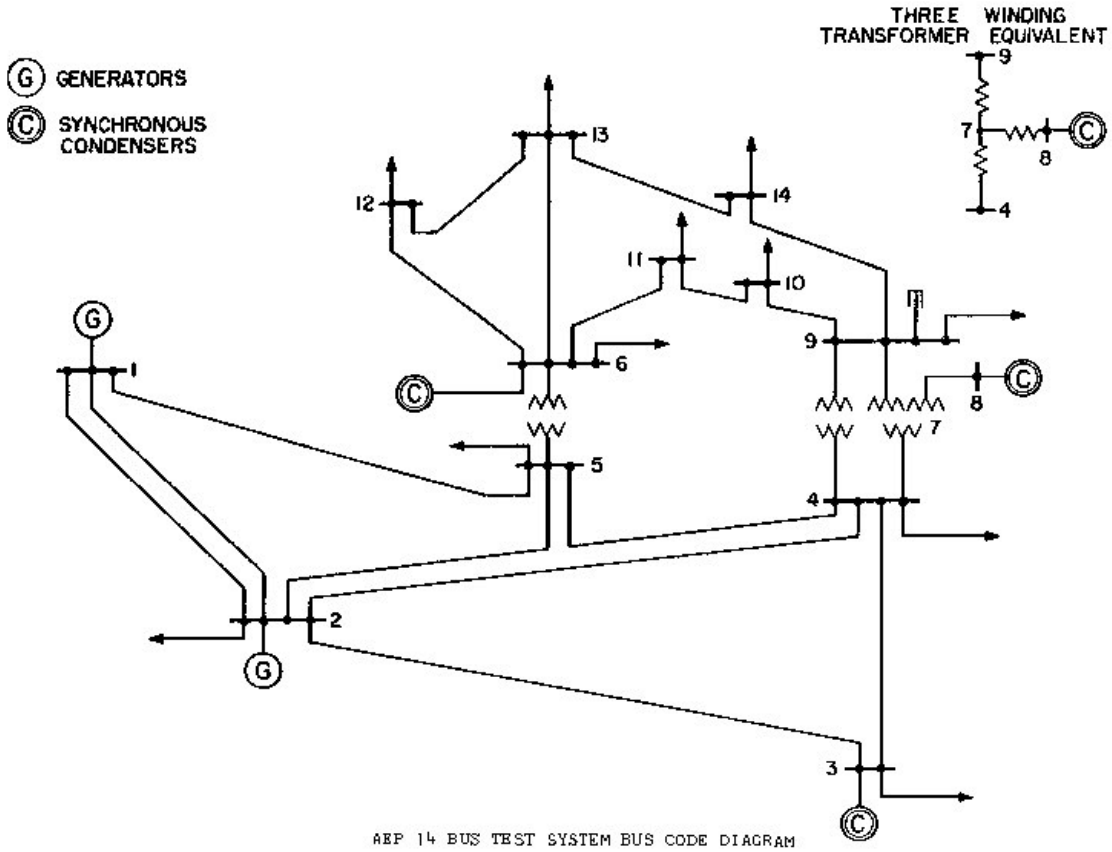

Figure 4.1: IEEE 14-Bus System (reprinted from [1])

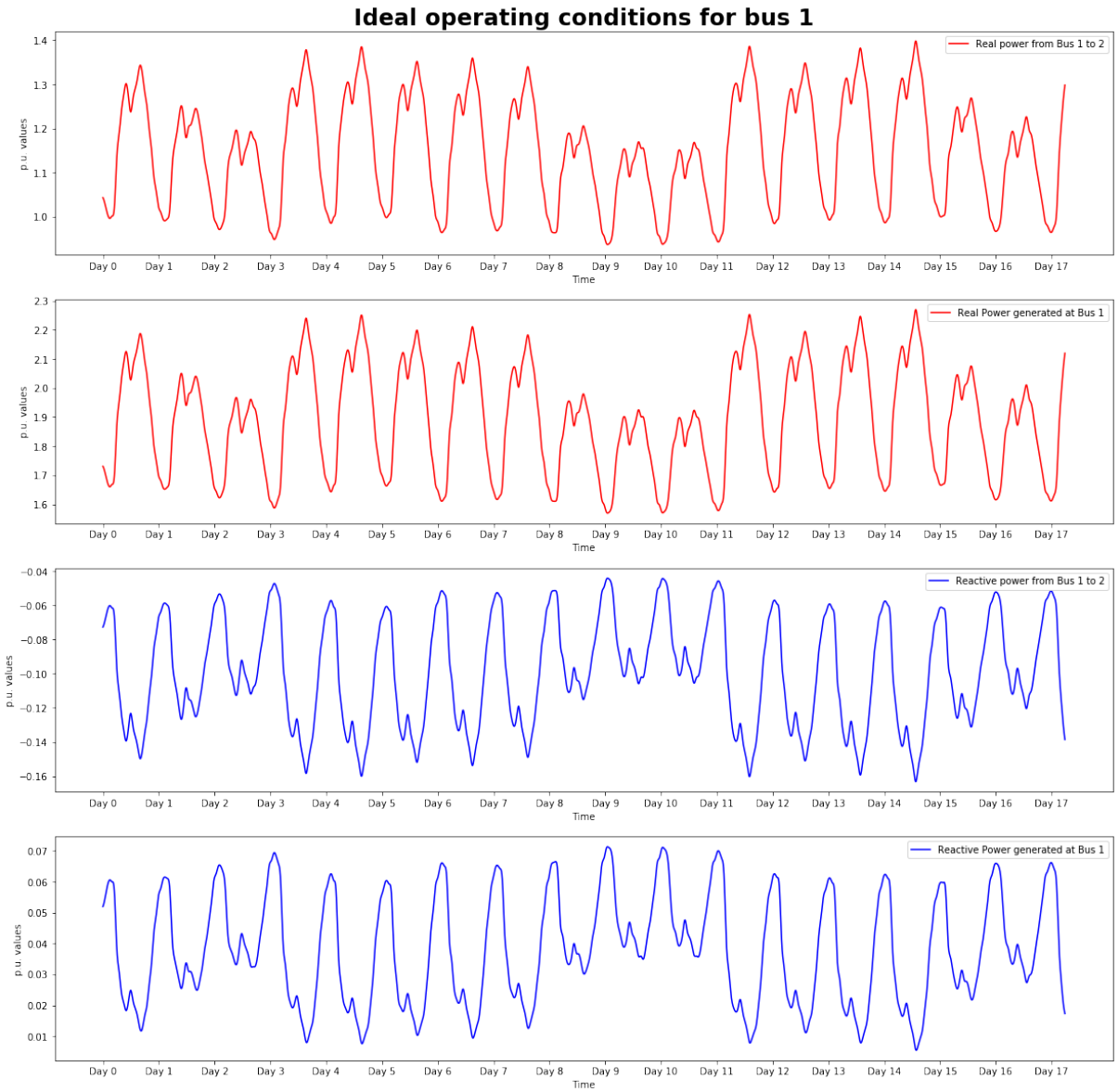

Figure 4.2: Ideal Grid Operating Conditions for Bus 1

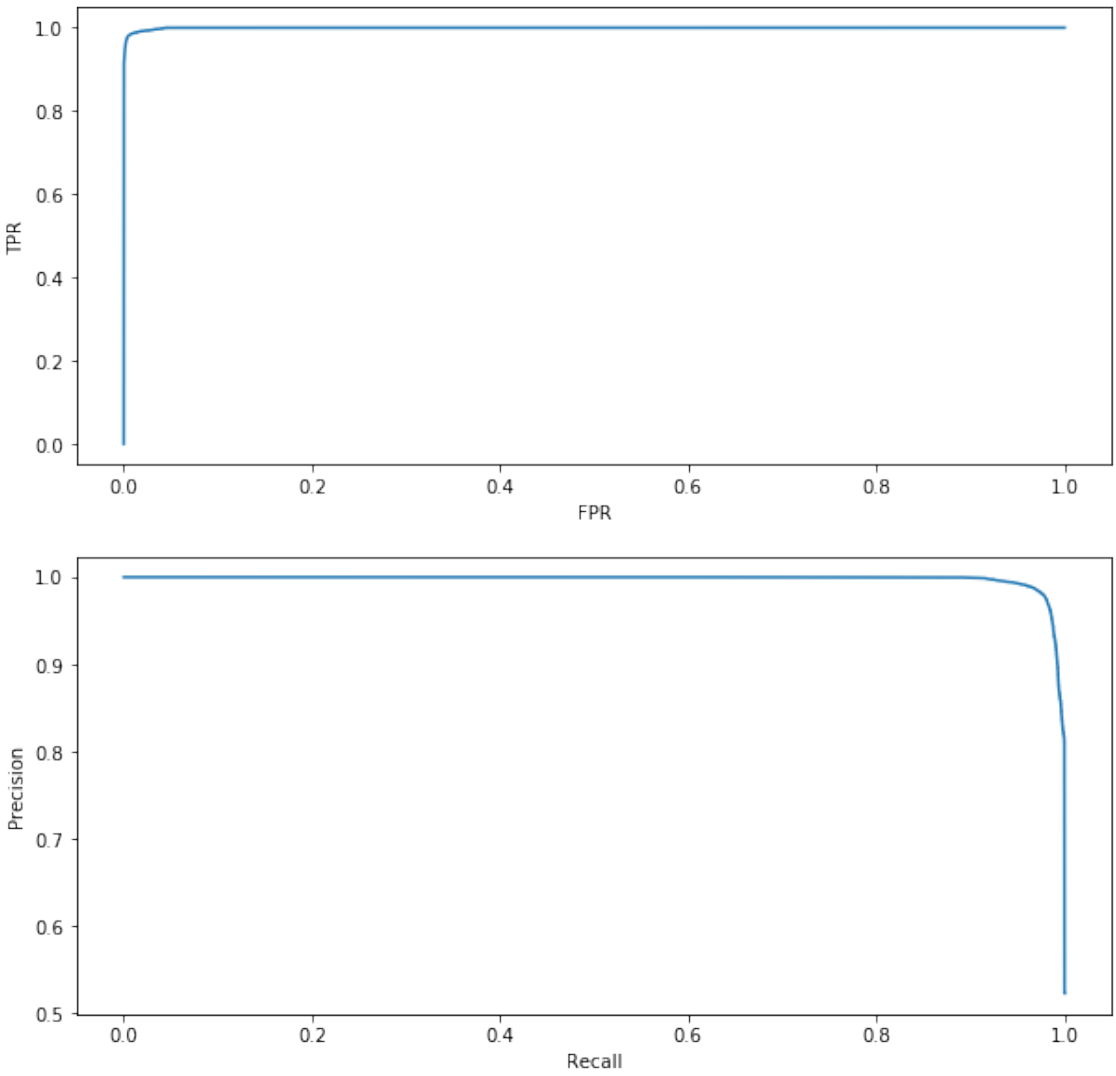

Figure 4.3: ROC Curve and Precision-Recall Curve

Number of devices compromised	4	8	12	16	20
ROC-AUC	0.9932	0.9902	1.0000	1.0000	1.0000
Recall	0.9940	0.9903	1.0000	1.0000	1.0000
Precision	0.9840	0.9807	1.0000	1.0000	1.0000
F1-Score	0.9890	1.0000	1.0000	1.0000	1.0000

Table 4.1: Detection using Fully Supervised Approach

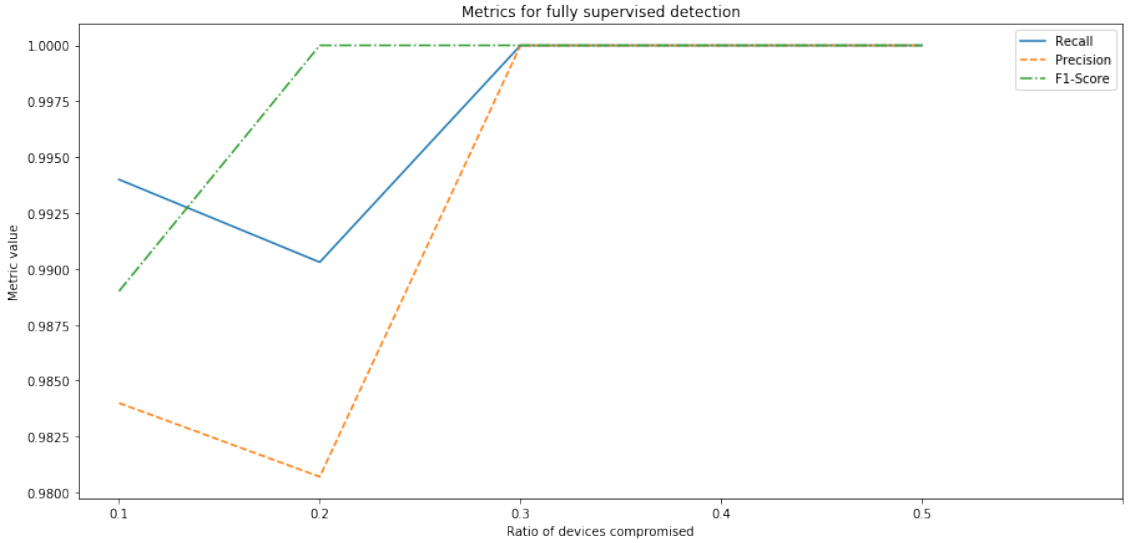

Figure 4.4: Metrics for Fully Supervised Detection

earlier, the output of the model gives the probability of an attack being present at the last time-step. But the threshold for classification needs to be calculated. This is done using the precision-recall curve (Figure 4.3), which we can tune based on our expectation of precision and recall from the model. Since we want our recall to be high, therefore we chose a recall threshold of 0.99. The results are shown in Table 4.1 and Figure 4.4. The unsupervised auto-encoder is trained on the entire baseline ideal grid operation data. A similar approach in storing model checkpoints is adopted as in the previous case. We also find the mean and standard deviation of total reconstruction error for the training data to help us to find the correct threshold for classifying the test set. At inference, we use the reconstruction loss from the auto-encoder and use multiple thresholds to plot the precision-recall curve to get an

Number of devices compromised	4	8	12	16	20
ROC-AUC	0.9114	0.9657	0.9998	0.9986	0.9992
Recall	0.8229	0.9313	0.9995	1.0000	1.0000
Precision	1.0000	1.0000	1.0000	0.9891	0.9944
F1-Score	0.9028	0.9644	0.9998	0.9945	0.9972

Table 4.2: Detection using Unsupervised Approach

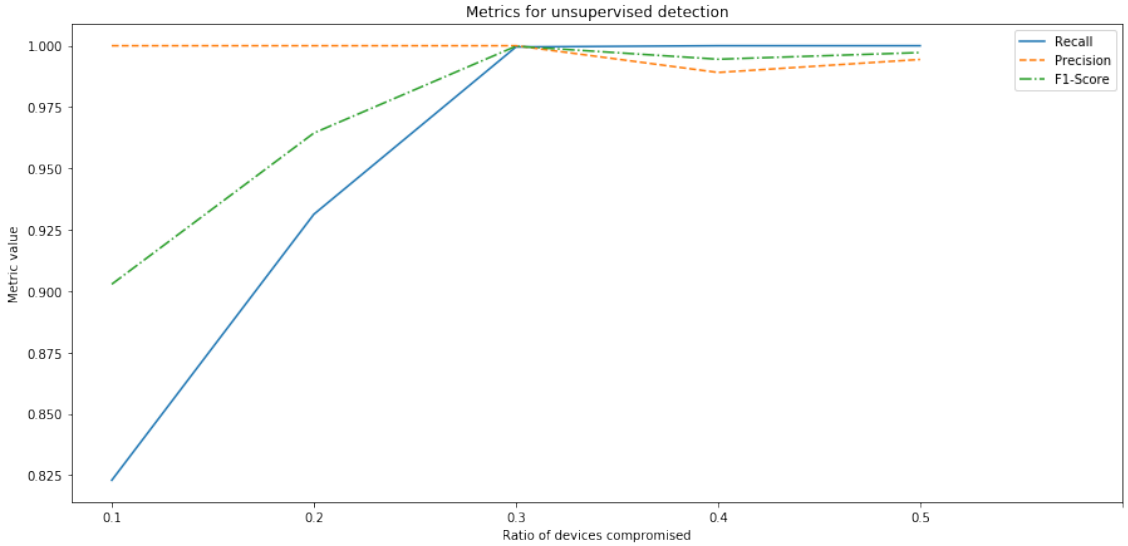

Figure 4.5: Metrics for Unsupervised Detection

optimal threshold of classification finally. The results are shown in Table 4.2 and Figure 4.5.

In [7] the authors mention a minimum number of devices that need to be compromised in order to successfully execute an attack with 100% probability. For our case that critical number is 14 which is more than 30% of devices compromised. It can be observed that both our detection methods can identify FDIAs when more than 30% of the devices are compromised with almost 100% accuracy. It can be observed that the unsupervised method does not perform as good as the supervised method when less number of devices are compromised. This might be because when less number of devices are compromised the attack vector generated does not cause significant deviations from expected behavior in the system, therefore

Number of devices compromised	4	8	12	16	20
ROC-AUC	0.8676	0.8811	0.8768	0.8815	0.8792
Recall	0.7357	0.7801	0.7636	0.8095	0.8129
Precision	0.7260	0.7036	0.5830	0.6629	0.7392
F1-Score	0.6819	0.7103	0.6491	0.6261	0.6648

Table 4.3: Location Specific to Device using Unsupervised Approach

reconstruction error is low. For cases where around 40-50% devices are compromised both methods perform perfectly. When compared to native error based detection methods(Recall = 0.1) both these methods outperform all of them. We have also compared our performance in detection with other methods which have tried before and have found ours significantly better, especially when more devices are compromised.

4.3 Location

Earlier we mentioned how the same unsupervised global detector could be used to find the specific devices that have been compromised. We use the same model trained for detection and find the mean and standard deviation of estimation error for every measurement separately on ideal operation data. We used multiples of the standard deviation to make the precision-recall curve and find the correct threshold of classification. While evaluating the performance of the model we use metrics for each measurement device separately and then take the median performance indicators as our final outputs. Taking median might be misleading if there are measurements which have not been affected by attacks in any case. Therefore, we eliminate all such meters which have not been attacked in test data points. The evaluation is individually run for all 250000 points for every level of intrusion, and the results are shown in Table 4.3 and Figure 4.6.

It can be seen that the performance of this model in precisely locating the attacked devices is not as good as in the case of detection. But we don't have a baseline to compare our performance because this is the first attempt at locating the devices compromised. As discussed earlier, locating exact devices might not be necessary for finding points or regions of

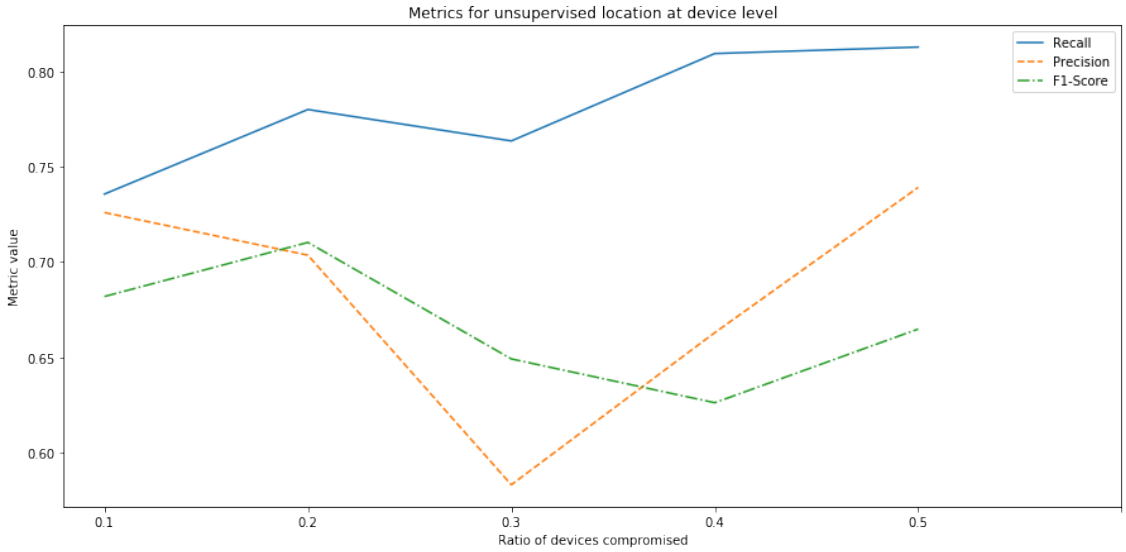

Figure 4.6: Metrics for Unsupervised Location Precise to Device

Number of devices compromised	**4**	**8**	**12**	**16**	**20**
ROC-AUC	0.9142	0.9173	0.9302	0.9184	0.8956
Recall	0.8286	0.8775	0.8647	0.8449	0.7987
Precision	0.9231	0.9236	0.9147	0.9205	0.9539
F1-Score	0.8733	0.8358	0.8890	0.8811	0.8694

Table 4.4: Clustered Location using Unsupervised Approach

intrusion because of the over-determined nature of the system. Therefore another alternative can be checking the performance of the model by clustering devices by the buses they are connected to. The performance results are shown in Table 4.4 and Figure 4.7. This method of clustering devices by the buses they are connected to helped to improve the performance of the auto-encoder by 10%.

Finally, we used the fully supervised case of locating compromised buses. We train individual models on each bus separately, which enables each of those models to be independent and therefore can adapt to local changes in network architecture. A critical thing to be noted here is that the model pertaining to a given bus uses measurements only from the devices connected to that bus. After training models for all the buses, we take the median score after

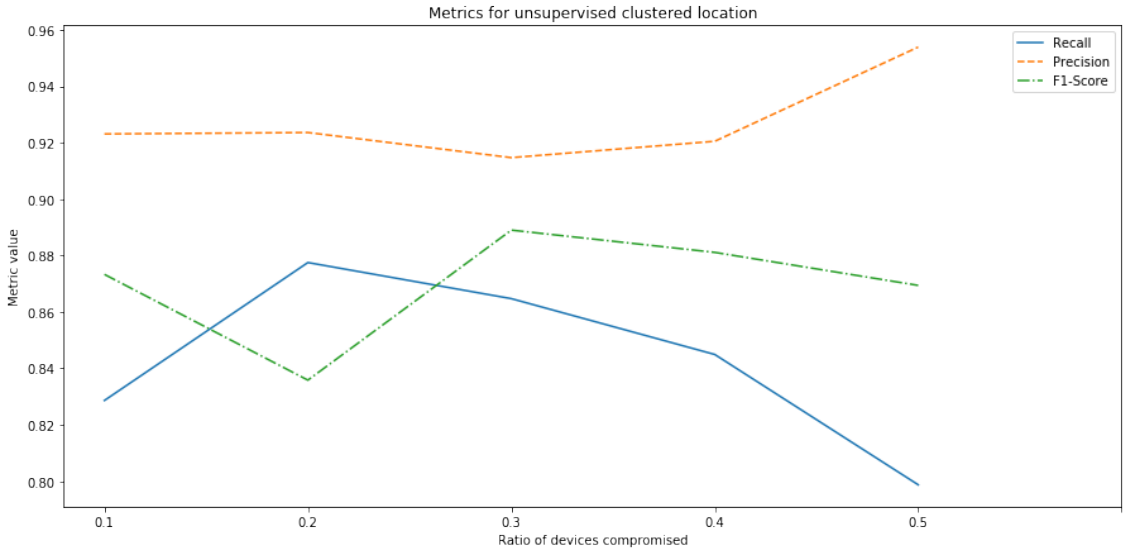

Figure 4.7: Metrics for Clustered Location using Unsupervised Approach

Number of devices compromised	**4**	**8**	**12**	**16**	**20**
ROC-AUC	0.9812	0.9927	0.9977	0.9991	0.9995
Recall	0.9405	0.9692	0.9819	0.9877	0.9964
Precision	0.8525	0.8500	0.8500	0.8500	0.8500
F1-Score	0.8943	0.9057	0.9112	0.9137	0.9174

Table 4.5: Clustered Location using Supervised Approach

eliminating the buses that are not affected at all in the test dataset. The metrics are shown in Table 4.5 and Figure 4.8.

It can be observed locating compromised devices is a more challenging task than detection of FDIAs. We can also observe that clustered location using the supervised approach is much better than that of the unsupervised approach. In addition as the number of devices grow the unsupervised approach start deteriorating in performance. This might be because the when larger number of devices are compromised there is a high probability that all devices connected to a given bus will be compromised. This will cause the reconstruction error to be low. The supervised model has been trained on cases where all the devices connected to a bus has been compromised, therefore the supervised method performs better than the

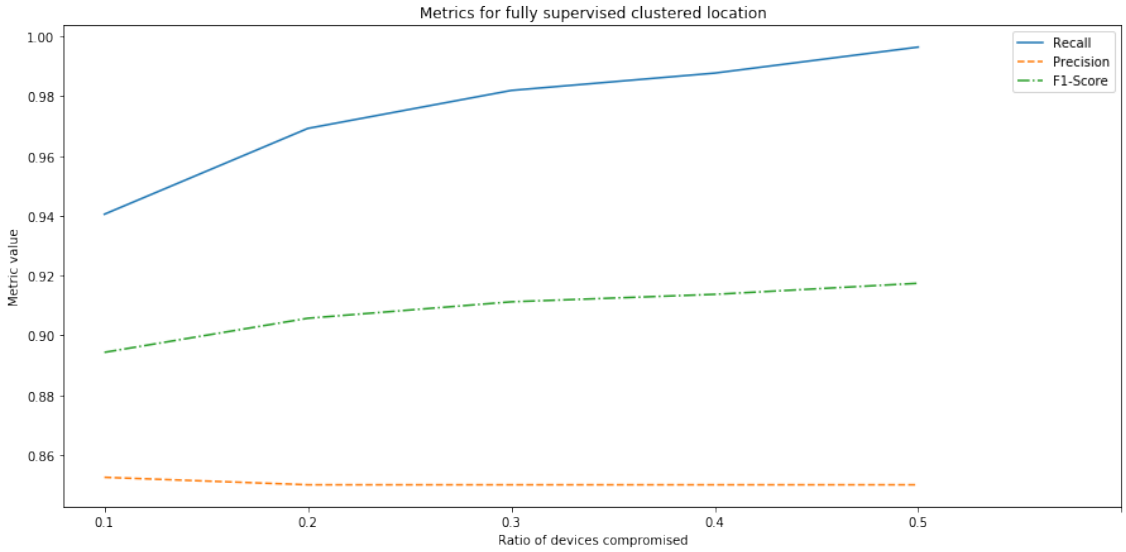

Figure 4.8: Metrics for Clustered Location using Supervised Approach

unsupervised model.

5. CONCLUSION AND FUTURE WORK

In this thesis, we discussed the application of deep neural networks for the identification of FDIAs in the electric power grid and the location of affected devices. We proposed a strictly supervised method of detection of attacks for which we simulated attacks on real-world power consumption data. We observed that the performance on test data was better than previously proposed methods by roughly 14% [53]. We also found that the performance of our model improved as more devices were compromised. We also noted the problems with this approach regarding the generation of training data for larger systems. Next, we proposed an unsupervised model based on a sequence to sequence auto-encoder which is trained on data representing ideal operating conditions of the grid. This model uses the spatio-temporal properties of grid measurement data of the past time-steps to predict the expected measurements for the current time-step. This novel idea does not need extensive training data and performs at par with the fully supervised case. We extended the same approach to locate the devices that are compromised and found that it performs reasonably well. We also found that although these methods were performing well, they might not adapt well to changing grid architectures and sizes. Therefore, we proposed a localized approach for detection and location of FDIAs which gives every bus to have its own model dependent on local measurement devices. This makes the system more sustainable. Besides, we also found that the median location performance was reasonable as well.

5.1 Immediate Extensions

The next step should be to test the proposed methodologies on larger power system cases. Another important study can be how model complexity is affected by the size of the power system. We can also check the performance of our models on contingency cases.

Another immediate extension can be using the auto-encoder on sub-grids or local patches of the grid. Since the system is over-determined, such local pools of measurement devices

can be created which can help to train an auto-encoder and use mutual information for correct re-construction as done for the global case.

5.2 Future Work

We can explore better attack generation algorithms using neural networks like Generative Adversarial Networks (GANs) so that the detector is robust to smarter attacks as well.

Another possible approach to anomaly detection in such spatiotemporally correlated data can be using convolutional neural networks (CNNs). The power grid network graph can be encoded in an adjacency matrix form, and the measurements can represent channels for every filled cell in the matrix. Since the adjacency matrix will be sparse, and a dense representation can be learned for this matrix which can further be decoded to the sparse form and the reconstruction error can be an estimate of intrusion.

REFERENCES

[1] D. Alert, "Analysis of the cyber attack on the ukrainian power grid," 2016.

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

[3] L. Streltsov, "The system of cybersecurity in ukraine: principles, actors, challenges, accomplishments," *European Journal for Security Research*, vol. 2, no. 2, pp. 147–184, 2017.

[4] Z. Zhang, S. Gong, A. D. Dimitrovski, and H. Li, "Time synchronization attack in smart grid: Impact and analysis," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 87–98, 2013.

[5] L. R. Phillips, B. Tejani, J. Margulies, J. L. Hills, B. T. Richardson, M. J. Baca, and L. Weiland, "Analysis of operations and cyber security policies for a system of co-operating flexible alternating current transmission system (facts) devices.," tech. rep., Sandia National Laboratories, 2005.

[6] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 4, pp. 1630–1638, 2017.

[7] 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*, vol. 14, pp. 1–33, May 2011.

[8] X. Liu and Z. Li, "Local topology attacks in smart grids," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2617–2626, 2017.

[9] G. Chaojun, P. Jirutitijaroen, and M. Motani, "Detecting false data injection attacks in ac state estimation," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2476–2483,

2015.

[10] Y. Zhou and Z. Miao, "Cyber attacks, detection and protection in smart grid state estimation," in *2016 North American Power Symposium (NAPS)*, pp. 1–6, IEEE, 2016.

[11] A. H. Yaacob, I. K. Tan, S. F. Chien, and H. K. Tan, "Arima based network anomaly detection," in *2010 Second International Conference on Communication Software and Networks*, pp. 205–209, IEEE, 2010.

[12] A. Ashok, M. Govindarasu, and J. Wang, "Cyber-physical attack-resilient wide-area monitoring, protection, and control for the power grid," *Proceedings of the IEEE*, vol. 105, no. 7, pp. 1389–1407, 2017.

[13] 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.

[14] P.-Y. Chen, S.-M. Cheng, and K.-C. Chen, "Smart attacks in smart grid communication networks," *IEEE Communications Magazine*, vol. 50, no. 8, pp. 24–29, 2012.

[15] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Sparse attack construction and state estimation in the smart grid: Centralized and distributed models," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1306–1318, 2013.

[16] L. Liu, M. Esmalifalak, Q. Ding, V. A. Emesih, and Z. Han, "Detecting false data injection attacks on power grid by sparse optimization," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 612–621, 2014.

[17] 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 Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1773–1786, 2016.

[18] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Smarter security in the smart grid," in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pp. 312–317, IEEE, 2012.

[19] Y. Huang, M. Esmalifalak, H. Nguyen, R. Zheng, Z. Han, H. Li, and L. Song, "Bad data injection in smart grid: attack and defense mechanisms," *IEEE Communications Magazine*, vol. 51, no. 1, pp. 27–33, 2013.

[20] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 203–233, 2008.

[21] R. B. Bobba, K. M. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. J. Overbye, "Detecting false data injection attacks on dc state estimation," in *Preprints of the First Workshop on Secure Control Systems, CPSWEEK*, vol. 2010, 2010.

[22] K. R. Davis, K. L. Morrow, R. Bobba, and E. Heine, "Power flow cyber attacks and perturbation-based defense," in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pp. 342–347, IEEE, 2012.

[23] K. L. Morrow, E. Heine, K. M. Rogers, R. B. Bobba, and T. J. Overbye, "Topology perturbation for detecting malicious data injection," in *2012 45th Hawaii International Conference on System Sciences*, pp. 2104–2113, IEEE, 2012.

[24] U. S. Goldstein M, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS ONE*, vol. 11, no. 4, 2016.

[25] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, pp. 41–46, 2001.

[26] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[27] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of artificial neural networks*. MIT press, 1997.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[29] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *arXiv preprint arXiv:1904.03288*, 2019.

[30] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[31] R. Chalapathy, E. Z. Borzeshi, and M. Piccardi, "An investigation of recurrent neural architectures for drug name recognition," *arXiv preprint arXiv:1609.07585*, 2016.

[32] D. Wulsin, J. Blanco, R. Mani, and B. Litt, "Semi-supervised anomaly detection for eeg waveforms using deep belief nets," *2010 Ninth International Conference on Machine Learning and Applications*, 2010.

[33] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[34] M. Ismail, M. Shahin, M. Shaaban, E. Serpedin, and K. Qaraqe, "Efficient detection of electricity theft cyber attacks in ami networks," in *2018 IEEE Wireless Communications and Networking Conference, WCNC 2018*, vol. 2018-April, pp. 1–6, Institute of Electrical and Electronics Engineers Inc., 6 2018.

[35] M. Nabil, M. Ismail, M. Mahmoud, M. Shahin, K. Qaraqe, and E. Serpedin, "Deep recurrent electricity theft detection in ami networks with random tuning of hyper-parameters," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 740–745, IEEE, 2018.

[36] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging*, pp. 146–157, Springer, 2017.

[37] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, pp. 1–18, 2015.

[38] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent," *Backpropagation: Theory, Architectures, and Applications*, vol. 433, 1995.

[39] 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.

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

[41] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

[43] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*, pp. 189–196, Springer, 2017.

[44] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

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

[46] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2837–2846, JMLR. org, 2017.

[47] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[48] C. D. Meyer and C. Meyer, *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, 2000.

[49] Q. Yang, J. Yang, W. Yu, D. An, N. Zhang, and W. Zhao, "On false data-injection attacks against power system state estimation: Modeling and countermeasures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 717–729, Mar. 2014.

[50] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, p. 167, June 1967.

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

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

[53] A. Ayad, H. E. Farag, A. Youssef, and E. F. El-Saadany, "Detection of false data injection attacks in smart grids using recurrent neural networks," in *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, IEEE, 2018.