The University of Sharjah
College of Engineering and College of Computing and Informatics
Department of Electrical Engineering and
Department of Computer Engineering

Senior Design Project Report

Phase 2 – Senior Design Project 2

AI-based FDI Countermeasure for IoE Smart Grids

Project Group:

U16200273	Abdalla Mohamed Alteneiji
U17102050	Abdalla H. Rashid Alshamsi
U16107097	Mohammed Emad Al-Haidary

Project Examination Committee

Supervisor	Dr. Ali Bou Nassif
Examiner 1	Prof. Tamer Rabie
Examiner 2	Prof. Mohamed Saad

Spring 2020/2021

Table of Contents

List of Figures iv
List of Tables v
Abstract 1
نبذة مختصرة 2
Problem Statement 3
1 Motivation 4
2 Introduction 5
2.1 Introduction to the Topic 5
2.2 Design Criteria 8
2.3 Our Design 8
3 Technical Background 10
3.1 Components of a Smart Grid 10
3.2 State Estimation 11
3.3 Bad Data Detection 12
3.4 Types of Attacks on Smart Grids 14
3.5 Why the FDI Attack? 15
3.6 Types of FDI Attacks 17
3.7 Why Target the Bypass BDD Attack? 18
4 FDI Attack 19
5 Alternative Solutions 20
5.1 Utilizing an RNN 20
5.2 Utilizing a CNN and an LSTM Network 21
5.3 Comparison 22
5.4 Conclusion 22
6 Proposed Design 23
6.1 Dataset Collection 24
6.2 Power System Simulation 26
6.3 State Estimation 27
6.4 Neural Network Structure 28
7 Design Implementation 30
7.1 Data Pre-Processing 31

7.2 System Simulation 31
7.2.1 Power Flow Analysis 31
7.2.2 State Estimation 32
7.2.3 FDI Attack Simulation 32
7.3 AI Training 33
Dataset Splitting 33
Data Windowing 34
Training Results 35
7.4 System Evaluation 35
Comparing Different Threshold Metrics 36
Evaluation Results 38
8 Conclusion and Future Work 39
9 List of Constraints 40
9.1 Environmental Concerns 40
9.2 Ethical and Social Issues 40
9.3 Economic Issues 40
10 List of Specifications 41
10.1 Parts List and Estimated Budget 41
10.2 Timeline 41
References 42
Appendices 44
1 Data Preprocessing 45
1.1 Code Excerpt 45
2 System Simulation 46
2.1 Power Flow Analysis Code Excerpt 46
2.2 State Estimation Code Excerpt 46
2.3 FDIA Simulation Code Excerpt 46
3 Training 47
3.1 Training Function Code 47
3.2 Data Window Generator Class Code Excerpt 47
3.3 Some Failed Training Attempts 48
4 System Evaluation 49
4.1 Code Excerpt 49

List of Figures

Figure 1 - Production vs. Consumption in Conventional Power Grids 7
Figure 2 - Production vs. Consumption in Smart Grids 7
Figure 3 - System Design Flowchart 8
Figure 4 - Main Components of a Smart Grid 10
Figure 5 - State Estimation Process 12
Figure 6 - State Estimation and Bad Data Detection 13
Figure 7 - Industries Affected by Cyber-Physical Attack on The Smart Grid 14
Figure 8 - Device Vulnerabilities in The Grid 15
Figure 9 - FDIA Impact on Affected Industries 16
Figure 10 - Popularity of FDI Amongst the Research Community 16
Figure 11 - FDIAs Classification 17
Figure 12 - General Architecture of an RNN 20
Figure 13 - System Design of Alternative Solution 21
Figure 14 - First Million Datapoints from The IHEPCDS Dataset 25
Figure 15 - IEEE 14 Bus Network 26
Figure 16 - Estimated Voltage Errors 27
Figure 17 - Estimated Phase Angle Errors 28
Figure 18 - Neural Network Structure 28
Figure 19 - Uses of Different Types of AI Models 29
Figure 20 - Measurment Vector Dataset 33
Figure 21- Data Windowing Operation 34
Figure 22 - Training Results 35
Figure 23 - Mean Absolute Error Plot 36
Figure 24 - Mean Absolute Squared Error Plot 37
Figure 25 - Mean Square Error Plot 37
Figure 26 - Senior Design Project 2 Timeline 41

List of Tables

Table 1 - Differences Between Traditional and Smart Power Grids 6
Table 2 - Comparison of Our Design with Alternative Solutions 22
Table 3 - Collection of Datasets Found 24
Table 4 – Data Preprocessing Final Results 31
Table 5 - Falsification Distribution 32

Abstract

Internet of Energy (IoE) describes a networked system of smart energy infrastructure components across generation units, loads, storage, energy meters, and automated distribution equipment. The core of IoE is the smart grid, which aims to detect, react and pro-act to changes in consumption and cyber-attacks. Smart grids, like any other digital system, are never perfect, and are hence vulnerable to attacks that could cause energy or financial loss, damage the system, or even destroy it. Amongst all the possible attacks on the smart grid, our focus in this project will be on the Fault Data Injection Attack (FDIA). The FDIA involves the adversary capturing measurements from different parts of the grid and falsifying them in such a way that they benefit from a cheaper bill for example, before sending the falsified measurements to continue to their original destination. We will focus in this project on the (Bad Data Detector) BDD-bypassing FDIA in particular as it is the stealthiest, most sophisticated and subsequently the most dangerous amongst all variants of the FDIA. BDD is a default component in any smart grid that detects anomalies and bad data across the grid. In this project, we will develop a detection system to detect BDD-bypassing FDIAs in a smart grid network using the field of deep learning as out primary tool. We will simulate the smart grid operation and the aforementioned FDIA before demonstrating the effectiveness of our detection system.

نبذة مختصرة

تصف انترنت الطاقة النظام الشبكي الخاص بمكونات البنية التحتية للطاقة الذكية عبر وحدات التوليد والتحميل والتخزين وعدادات الطاقة ومعدات التوزيع الآلي. تعتبر الشبكات الكهربائية الذكية محور انترنت الطاقة، حيث تهدف إلى رصد التغيرات في الاستهلاك والهجمات الالكترونية والتفاعل معها والاستجابة لها. إلا أن الشبكات الكهربائية الذكية، وكغيرها من الأنظمة الذكية، ليست مثالية وبالتالي فهي عرضة لهجمات إلكترونية قد تسبب خسارة طاقية أو مالية أو قد تلحق ضررا بالنظام أو تدمره. في هذا المشروع سوف نركز من بين كل الهجمات المحتملة على الشبكات الكهربائية الذكية على هجمات إدخال البيانات الخاطئة. تتضمن هجمات ادخال البيانات الخاطئة التقاط القياسات من أجزاء مختلفة من الشبكة وتزييفها بطريقة تستفيد من فاتورة أرخص على سبيل المثال، قبل إرسال القياسات المزيفة للاستمرار في وجهتها الأصلية. سنركز في هذا المشروع على الهجمات التي تتعدى كاشف البيانات السيئة بالتحديد، بسبب كونها الأشرس والأكثر تعقيداً ايضاً، و بسبب كون كاشف البيانات السيئة جهاز يتواجد في جميع الشبكات الكهربائية الذكية، و يقوم الكاشف بالكشف عن أي بيانات مشكوك بها في نطاق الشبكة. في هذا المشروع، سوف نقوم ببناء نظام تحري لهذه الهجمات التي تتعدى كاشف البيانات السيئة بمساعدة التعليم العميق كنظام أساسي. سوف نقوم بتجربة هذه الهجمات لإثبات نظامنا الكاشف عن هذه الهجمات.

Problem Statement

The purpose of this project is to design and develop to the current countermeasures against one of the most lethal attacks against an IoE smart grid, the FDI attack. Our aim is to make smart grid technology safer and more viable for our future. Smart grids are the beating heart to today's homes and infrastructure. It is therefore of high interest for adversaries, which makes its cybersecurity systems paramount for national security and the high availability of electricity. Any loopholes can make the system vulnerable to numerous attacks, each of which would have serious consequences, ranging from the leakage of customer data, to billing fraud, to a catastrophic cascade of system failures. Time has proved that, no matter how safe those technologies look, they are still vulnerable, and in risk of software attacks. They face a huge number of software attacks that are difficult to detect, and they require complicated approaches to detect and prevent. Protecting this zone, which consists of nuclear, thermal, and hydroelectric power plants, from any type of attack assures us a safe and consistent distribution. This allows us to maintain a steady equalization between the levels of production and consumption of energy.

1 Motivation

Electric power supply is a crucial factor for lots of areas, including manufacturing, healthcare, education, water distribution, residential areas, and transportation systems, many of people lives rely on being supplied with electricity, people in hospitals living on ventilators could lose their lives if the power system is under threat, other people could lose their jobs if power supply is not satisfying their demands.

Since the cybersecurity state of any IoE has high stakes, we are determined to take the largest threat in the field and develop a contributing countermeasure to fortify the currently existing security measures.

Moreover, any interruption in electric power is likely to have an unwanted impact on the whole residential or commercial network. Within the last decade or two, the huge growth of smart grids, that have replaced conventional grids that seem to not be fulfilling the needs of both suppliers and consumers, has revealed huge importance to keep an eye not only on the traditional grid problems, but also support the development of power generation from renewable sources. Furthermore, since power suppliers are obligated to meet customers demand during peak hours, they should try to balance, or maintain a specific way of distribution that assures that there is not any type of squandering.

2 Introduction

This chapter introduces the reader to the origins of the smart grid and their differences to the traditional power grid, which will enable them to have a clearer perspective towards the motivation and the reasoning behind our selection to this particular project objective.

2.1 Introduction to the Topic

Smart grids and conventional power grids vary in many areas, conventional grids are one directional grids that provide energy from the distributor to the consumer with no extra analyses that could help service providers to know and track the usage. Smart grids on the other hand are two directional and helps save energy since the distributor could have an entire view on all vital readings that could help on focusing on loss and waste of energy, this is because of components that are used in smart grids, such as sensors, remote monitoring, automated switches, capacitor banks, and many more equipment that a smart meter is equipped with. Another major difference between a smart grid and a conventional grid is that if smart grids encounter any damage, smart grids are adaptive and self-healing since they could process this harm internally and in real-time, in contrast, conventional grids are more sensitive and delicate, the damages could affect an entire city, and shall be dealt with from an outer source such as a professional technicians which is much more costly. Another important factor is the customer choice; in smart grids, customers have more options in the types of services they could acquire making it more flexible and controllable, but in conventional grids, choices are restricted to that almost everything is built-in. In addition, conventional power grids occupy more space and they are huge in size, but smart grids are smaller and easier to install.

In the below table are few summarized differences between the traditional power grids and smart grids:

Characteristics	Conventional power grids	Smart grids
Technology	Electromechanical, in the sense that it is built in with a mechanical device that is electrically operated.	Digital, which means they are employed with digital technologies that allows for increased communication between devices.
Distribution	One-way distribution, where power is only transferred from the plant to the customer using traditional techniques.	Two-way distribution, where power is transferred in a smart way that assures no leakage or loss.
Generation	Centralized, where all power is generated from a central location, which does not allow for easy incorporation for alternative power sources	Distributed, where power can be distributed from different power plants, and alternative solutions are in hand.
Sensors	Few sensors, where the infrastructure cannot occupy lots of sensors.	Sensors throughout, where sensors are available everywhere in smart grids to measure and send many important readings.
Monitoring	Manually monitored, where energy distributions must be monitored physically.	Self-monitored, where the smart grid is able to balance power loads and manage distribution without the need for any external interference.
Restoration	Manual, where technicians are needed to make repairs on the grid.	Self-healing, where the smart grid can identify and troubleshoot without technicians.
Control	Limited, where companies cannot control energy after leaving the power plant.	Pervasive, where energy and energy consumption can be controlled due to the availability of sensors.

Table 1 - Differences Between Traditional and Smart Power Grids

Performance in conventional grids and smart grids varies a lot in the sense of both production and consumption. In conventional grids, production is always constant no matter what the consumption is, meanwhile in smart grids, if consumption decreases, this decrease is read and responded to with a decrease in production with the goal of conserving and better distributing power to the rest of the grid. Below are Figures 1 and 2 demonstrating a short comparison between the production and consumption in both conventional power grids and smart grids.

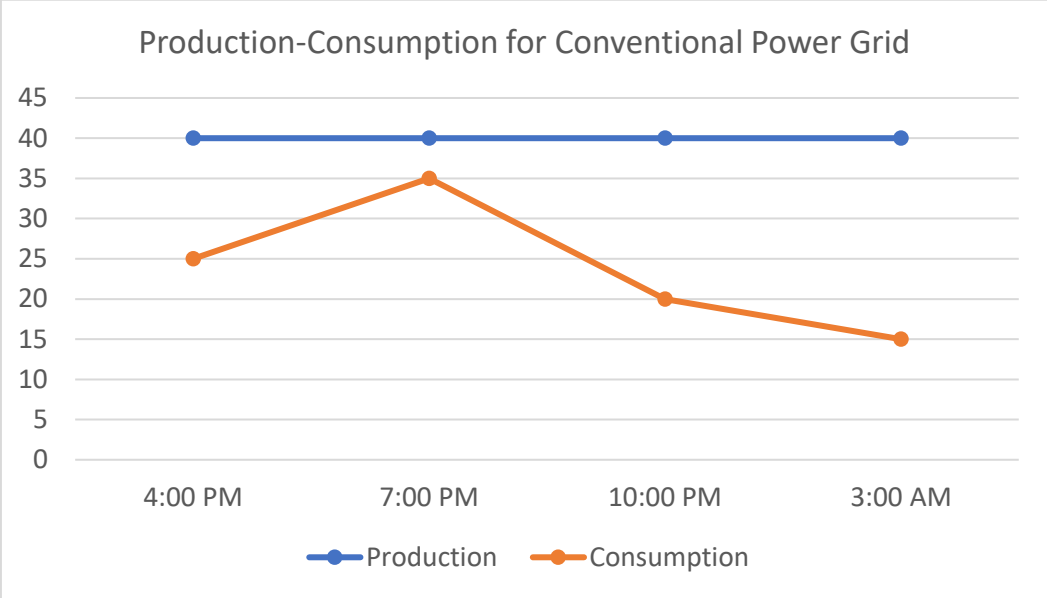


Figure 1 - Production vs. Consumption in Conventional Power Grids

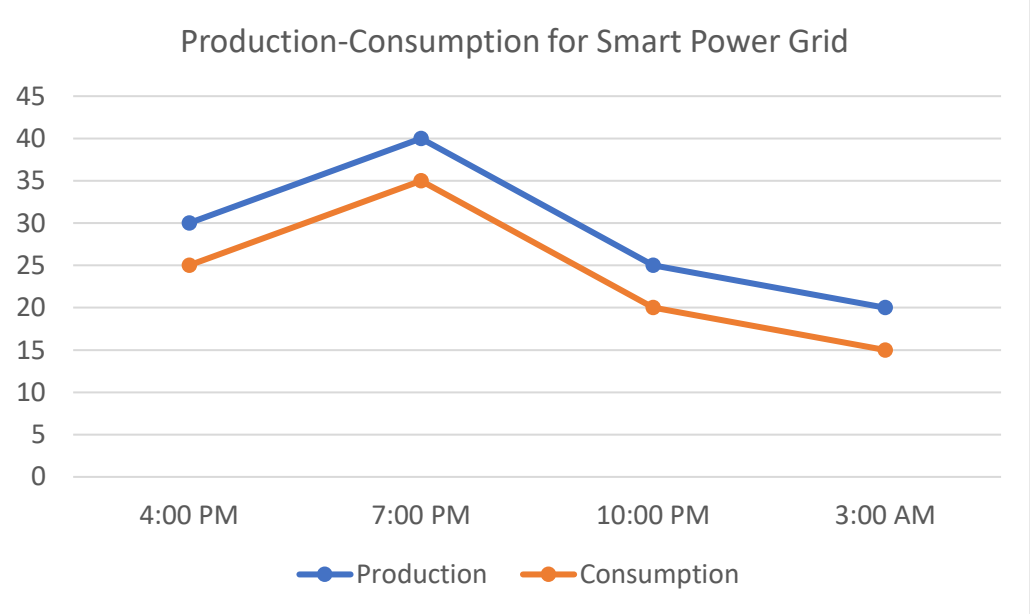


Figure 2 - Production vs. Consumption in Smart Grids

2.2 Design Criteria

Since the core of our proposed countermeasure will be based on Artificial Intelligence (AI), we have come up with a set of criteria that needs to be present in our system design in order for us to call it a successful project.

The criteria we have set are accuracy, ability to detect false positives, scalability and reliability.

- **Accuracy:** The accuracy of the AI model to correctly detect the occurrence of an FDI attack is the center selling point of our design.
- **Scalability:** The ability of the countermeasure to handle datasets smaller or larger than the size of the dataset(s) used to train the AI model.
- **Reliability:** How stable is the output of our countermeasure will determine its reliability and hence its viability to be used in practical applications.
- **Real time operation:** detection system should not need knowledge of any future measurement.

2.3 Our Design

Figure 3 shows the components making up our system design. The upper part is the attack generation and the smart power grid's loads and control center. The lower part is our system design which is our detection system that includes our AI model.

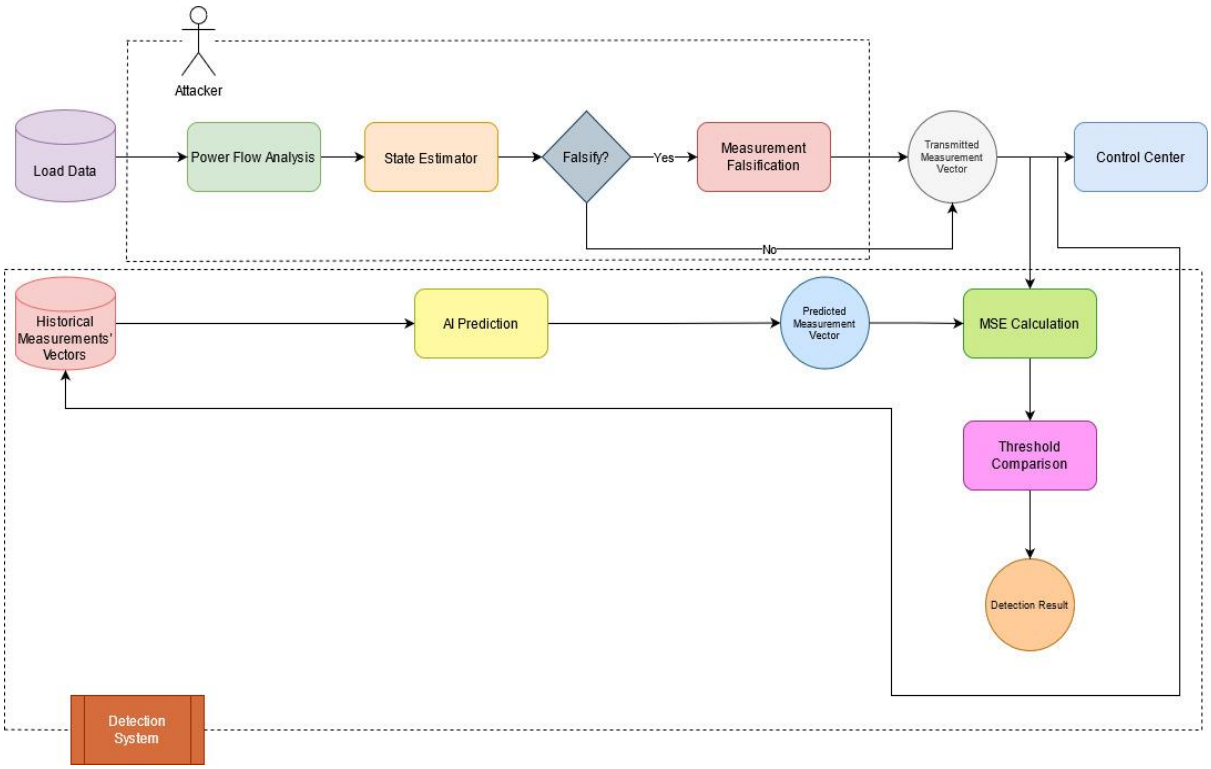


Figure 3 - System Design Flowchart

The theoretical concept of the system's mechanism is that the FDI generation (represented by the attacker) is going to bypass the bad data detector (BDD). The main purpose of BDD is finding any falsified data that inject before the measurement vector reaches the control center. Our system which contains our AI model is going to predict the next vector with a high accuracy. Then MSE is calculated between the predicted vector and the received measurement vector. The MSE will then be compared with a pre-selected threshold, and if the MSE calculated exceeds the threshold, we say an FDI attack has been detected.

3 Technical Background

3.1 Components of a Smart Grid

A smart grid consists of many components and elements, which are distributed in three parts: generation, distribution, and consumption.

In generation, there exits an element called SCADA, or Supervisory Control and Data Acquisition (SCADA) systems, which allow the utility to remotely monitor and control network devices as a means of achieving reliability and demand efficiencies for the utility as a whole. Another important component that is based on the generation is the Phasor Measurement Unit (PMU). PMUs are measurement devices that measures voltages and currents in phasors. Another component in smart grids are FACTS, which stands for Flexible AC transmission system devices, those devices are responsible for system stability, midpoint voltage support, and reactive power control in grid interconnections. FACTS are able to do a significant number of tasks; FACTS devices are able to enhance the power transfer capacity between available transmission lines without building a new line, drop the impedance in a transmission line, and improve the power factor due to its ability to increase the phase angle.

In distribution, normal transmission lines are used to distribute energy from production to consumers.

In consumption, or in customers' locations, smart meters are installed, smart meters are electronic devices that records important data such as consumption of power, voltage levels, current, and power factor. They send information to the consumer for a better view of consumption behavior and sends them to energy distributors for system observation and customer billing. They usually record energy near real-time, and usually reports short intervals during the day.

Figure 4 below is a depiction of the three main sub-systems that make up a smart gird [1]:

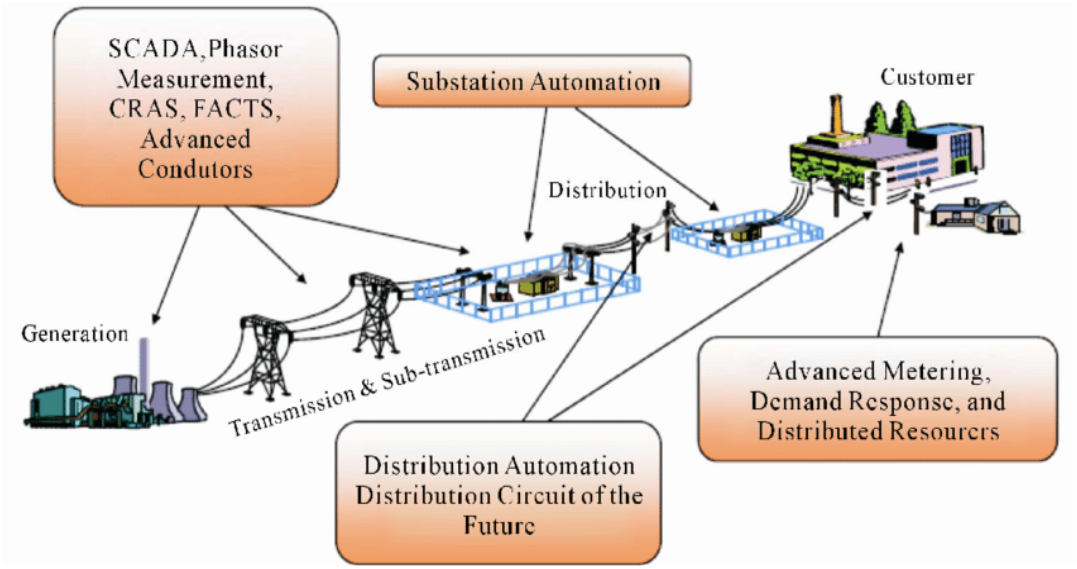


Figure 4 - Main Components of a Smart Grid

3.2 State Estimation

State estimation algorithm is commonly based on the weighted least square method, where the state variables, which are the voltage magnitude and phase angle, are determined by the minimization of the square of the error of all measurements within a chosen time period. Estimates or approximations are used in smart grids due to several reasons, one is that smart meters are not always perfect, and they are not available everywhere, which makes estimation a crucial part to suppress any uncertainties in the measurements.

State estimators can be either static or dynamic, static state estimators in a power system are built on the assumption that the state variables which are the voltages and phase angles are in steady state, or not changing within a time interval, while in dynamic state estimation the model is built on the assumption that the state variables are of changing behavior within a time interval.

State estimations vary in different way, each has a different mathematical model and calculation the estimations using a different approach. The method that we will implement in estimation is the least square estimation method, which means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation, the least square method can be expressed mathematically in the following equation:

$$z = h(x) + e \tag{1}$$

Where z is the measurement vector, x is the state variable vector, h is the nonlinear power flow equations, and e is the measurement error vector.

Practically, to attack a specific state variable, we should replace all measurements that the state estimation depends on to falsified ones.

In addition, our model will be a DC power-flow model, meaning that our simulated system will be DC based, and based on that, the above equation can be further simplified to the following equation:

$$z = Hx + e \tag{2}$$

Where z is the measurement vector, x is the state vector, H is the topology-dependent Jacobian matrix, and e is the measurement vector.

3.3 Bad Data Detection

Bad data detection (BDD) is one of the important steps in the sequence of procedures in state estimation, bad data refers to inaccurate or falsified data.

Bad data detection is used to detect errors and false readings in a conventional power grid, but as studies have shown, BDD failed to detect falsified datapoints caused by FDI. This created a huge need for a countermeasure that would lead us to come up with our system design for possible FDI attacks onto a smart grid.

One of the types of FDI attacks comes from the Bad Data Detection, which is called the Bypass BDD, this attack aims to bypass and avoid the Bad Data Detection algorithm directly.

The main threat smart grids and most power grids encounter is energy theft, energy theft is the most common reason why adversaries inject fault data into the power system, which leads to incorrect readings, or even device breakdown in the power generation. Energy theft causes huge economic loss in many energy departments across the globe, since energy theft causes energy that is distributed not to be billed on the consumer. It can be undertaken by meter tampering. Energy theft will be discussed in greater details in the upcoming parts.

Figures 5 and 6 explains how BDD is directly linked to state estimation and how crucial it is in the process of filtering or eliminating bad data [3]:

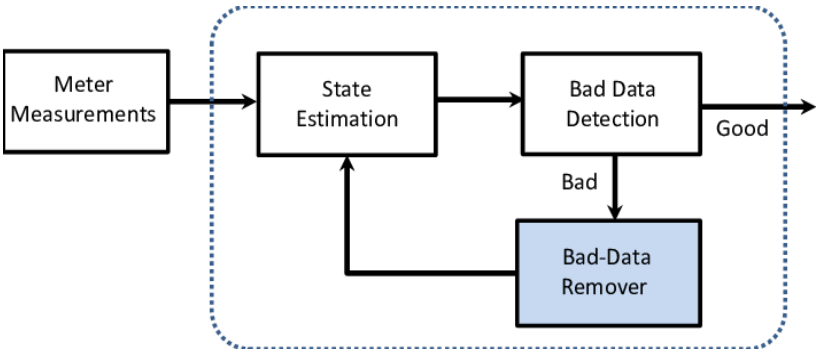


Figure 5 - State Estimation Process

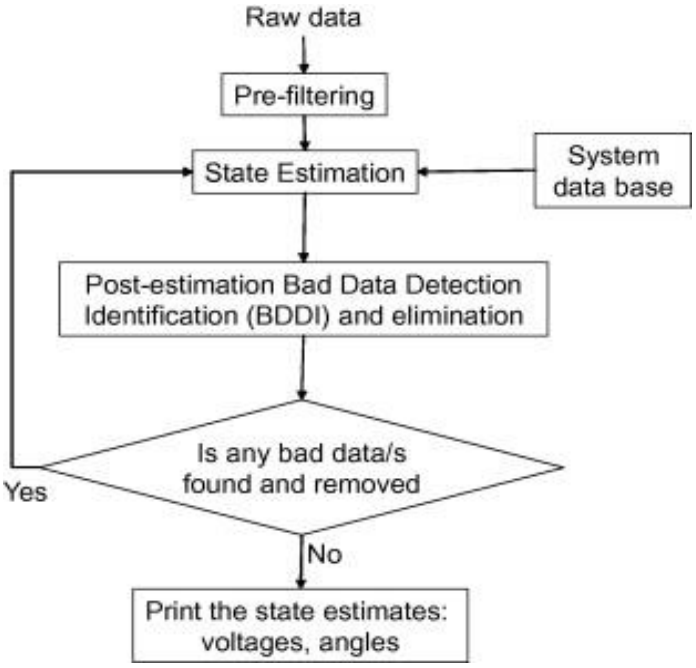


Figure 6 - State Estimation and Bad Data Detection

It is important to note that a bad data detector has a certain threshold, this threshold, τd, is set to identify bad measurements from a set of given measurements, and to mathematically understand if the given measurement exceeds the threshold or not, the below equation is found:

$$||\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}||_2 \geq \tau_d \tag{3}$$

Where z is the measurement vector, x^ is the Estimate state vector, H is the topology-dependent Jacobian matrix, and τd is the appropriate threshold. If the absolute value of the subtraction exceeds the threshold τd, or even equal to it, the set of measurements is suspected to be "bad data".

It is important to note that because BDD highly relies on state estimation, the FDI attack is not detected since it bypasses the state estimator.

3.4 Types of Attacks on Smart Grids

Attacks in smart grid is a combination of cyber side (for network intrusion and data injection) and physical side (for the bad data construction). In this section, the related methods for the FDI attack are summarized.

1. **Cyber side:** Techniques on the cyber side are the foundation of the FDI attack. The basic target for cyberattack is to obtain the loophole to make invalid operations on smart meters or network communication without authorization.
2. **Physical side:** With the cyberattack intruding into the smart grid network, attackers can change the meter readings and feed tampered data back to the control center. When attackers are armed with the knowledge of power system configurations, they can construct bad data according to the physical model of smart grid, bypassing the bad data detection.

Figure 7 shows the different affected industries in the event of a cyber-physical attack on the smart grid [4]:

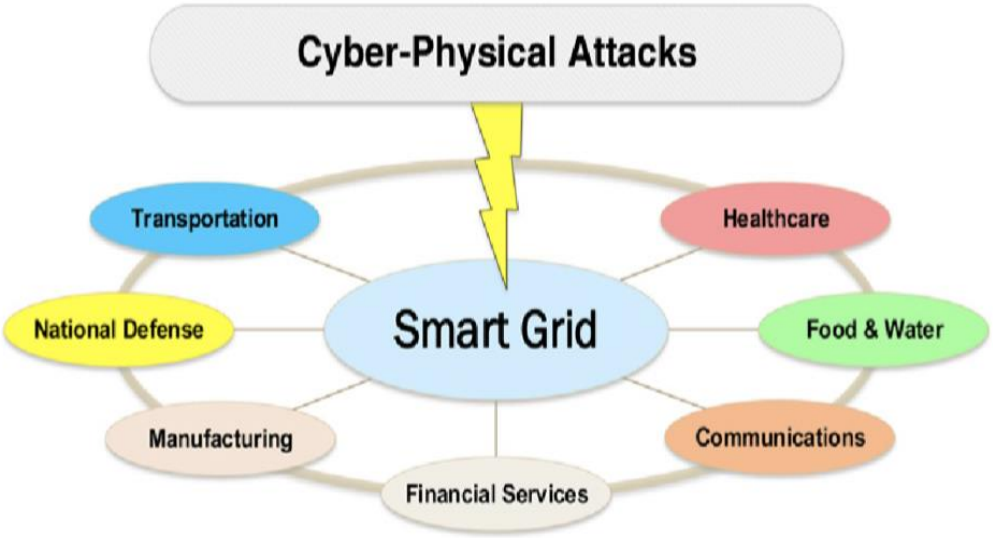


Figure 7 - Industries Affected by Cyber-Physical Attack on The Smart Grid

Any infrastructure is vulnerable to attack. Whether the vulnerability is great or small is determined by the mitigation and security techniques implemented around and within it. In the smart grid, specific elements and security requirements are necessary for operation.

Vulnerabilities can be described and categorized in many ways. We can view the weaknesses of the smart grid on a device or entity basis or as a combination of those entities. A list of devices that have specific vulnerabilities and important purposes on the grid is listed in Figure 8 [5]:

Operational Systems	IT Systems	Communication Protocols	Endpoints	Human Factors
Generators	PCs	Wifi (IP)	Electric Vehicles	Human Training
Transformers	Servers	Zigbee	Smart Meters	Social Engineering
SCADA	Apps	4G	Mobile Devices	Phishing
PMU	DBs	DNP3	IEDs	Data Transfer
PLC	Web Services	IEC 60870		
Smart Meters		IEC 61850		

Figure 8 - Device Vulnerabilities in The Grid

Most of the effective attacks affecting the smart grid are a combination of several of the vulnerable entities attached to it. Whether the goal is malicious or for testing purposes, normally the exploitation of highly valuable resources employing security mechanisms requires complicated procedure to complete. This could consist of a coordinated attack carried out in a distributed fashion and utilizing several different types of technology and attack vectors.

3.5 Why the FDI Attack?

False Data Injection Attack (FDIA) is one of the most dangerous attacks. It aims to inject false data to launch a judicious attack that can cause severe damages.

The attackers inject malicious packets into the targeted network to disrupt network services, by either compromising the sensor nodes or hijacking the communication channel.

In [6], the authors presented two FDIA attack scenarios in which they can successfully inject malicious data to the state estimation in smart grid. Unfortunately, these attacks are confirmed by real-world incidents like the Ukrainian electric power blackout at- tack in December 2015, which affected approximately 225000 customers in Ukraine for several hours by remotely opening the substation breakers.

Figure 9 shows the impacts of FDIA on each affected industry [7]:

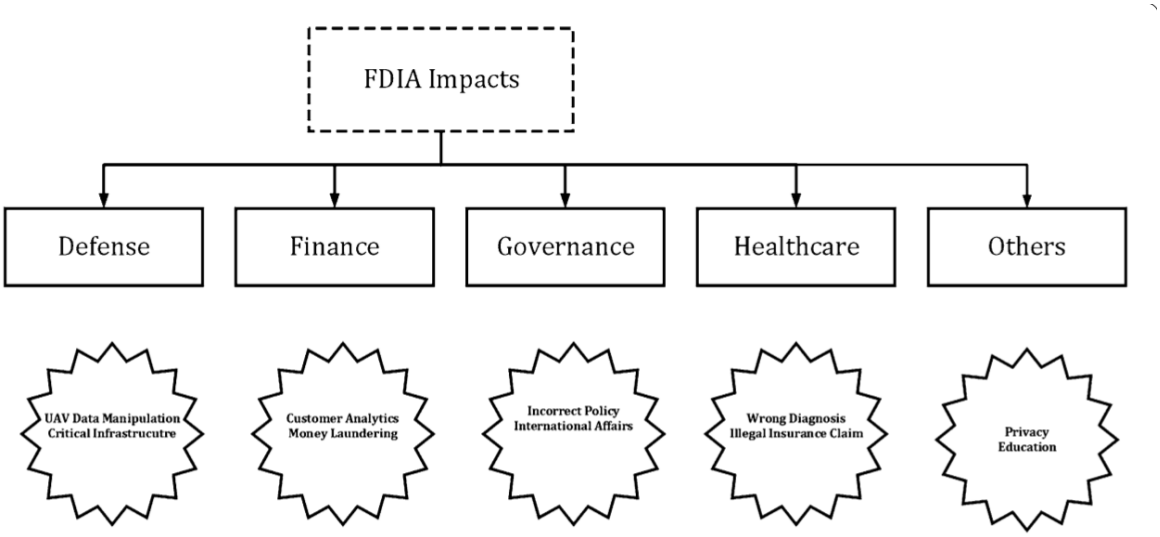


Figure 9 - FDIA Impact on Affected Industries

It is noteworthy to point out how an FDI attack could damage critical defense infrastructure, which would make this possibly a matter of national security.

The primary reason why we chose to work on the FDI attack is due to the fact that it has drawn most attention from the research community as shown in Figure 10 [8]:

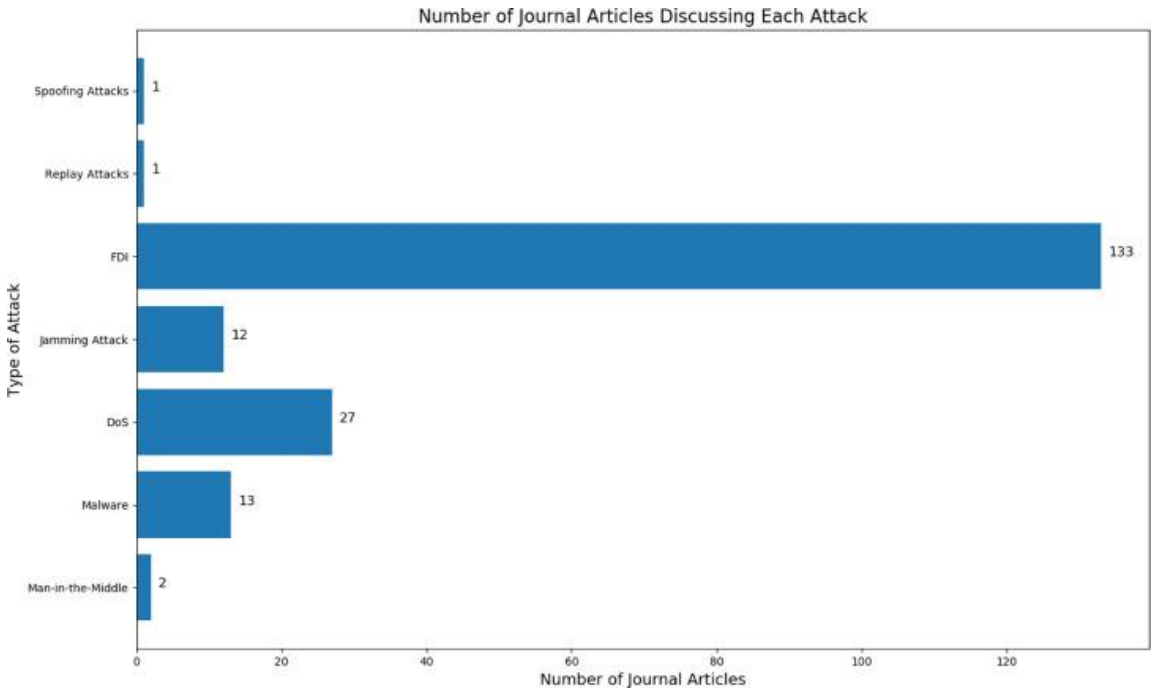


Figure 10 - Popularity of FDI Amongst the Research Community

3.6 Types of FDI Attacks

There are different modules that can be targeted by an attacker to damage the smart power system: state estimation, contingency analysis, SCOPF, load forecasting and SCED. In addition, DSSE.

As shown in Figure 11, the proposed three-level classification, which categorizes the FIDAs with respect to the targeted systems at the first level [6]:

Three-level classification of FDIA.

System	Subsystem	Attack	Reference
EMS	State estimation	Bypass BDD	[10,15–20]
		Observability	[21–23]
		Topology	[24–32]
	Contingency analysis	CA	[13]
		OPF	[33,34]
	SCOPF	SCOPF	[35]
		AGC	[36–39]
MMS	SCED	SCED	[11,12,14,40–45]
DS	Volt/var control	Volt/var control	[46–48]
	Price determination	Energy theft	[49,98–100]

Figure 11 - FDIAs Classification

We briefly explain the different types FDIAs [6]:

- **Bypass BDD:** Bypass the Conventional Bad Data Detection Algorithm directly.
- **Observability Attack:** Aims to make the system unobservable. This is achieved by injecting a sophisticated vector attack to confuse the control center and make it unable to distinguish between network unobservability and presence of malicious attack. It is known as the Stealthy FDIA.
- **Topology Attack:** Attackers can inject false data about the logic measurements, affecting the SE by the control center.
- **Contingency Analysis Attack:** A malicious attacker can target the Contingency Analysis by either adding or removing a contingency element to the contingency list that is constructed by the CS.
- **Optimal Power Flow (OPF) Attack:** OPF optimizes the power dispatch of the power system network considering objectives such as total generation cost and system loss.
- **Security-Constrained Optimal Power Flow (SCOPF) Attack:** SCOPF is the system responsible if a contingency analysis produces a positive result. An attacker can compromise the operations of the SCOPF, causing a cascading failure.
- **Automatic Generation Control (AGC) Attack:** Adjusts the output power of different generators in neighboring areas depending on the load change. It is carried out every 5 seconds compared to the SE which is carried out every 5 minutes.

- **Security-constrained economic dispatch (SCED) Attack:** Responsible for determining the amount of energy produced by each generator.
- **Volt/var control (VVC):** Aims to keep the voltage of the network in a nominal range, which is useful to stabilize the network load.
- **Energy Theft:**
    - **Individual Attack:** On a single smart meter.
    - **Collusive Attack:** A smart meters in pairs.
    - **Massive Tampering Attack:** On a large scale of any number of smart meters.

3.7 Why Target the Bypass BDD Attack?

BDD highly relies on state estimation, so in order to detect bad data in a bad data detector, the state estimator is the judge. The dependency on state estimation is what makes a bad data detector vulnerable to FDI attacks, if an attacker has access to the state estimator, and can recalculate the values of the Jacobian matrix, which is a part of the state estimator formula, the attacker can easily fool the system resulting in an undetected FDI attack. In order to know the Jacobian matrix, the attacker should have an idea of the network topology, or structure of the network under attack. In addition, the attacker can have access to all measurement vectors, allowing the attacker to calculate the corresponding error vectors, which both require the attacker's knowledge of the network topology or structure.

To simulate this attack and build a model, we assume the attacker has access to all mentioned elements and has the maximum knowledge of the network topology and knows the measurement and error vector. If an attacker has the required knowledge, an attacker can bypass the state estimator, and simultaneously bypass the BDD, and fool it to result in an undetected FDI attack.

4 FDI Attack

Our simulation of this attack was network-wide to ensure that FDI simulation was successful.

Attack can be expressed in equation (4).

$$\boldsymbol{z_a = z + a + e} \tag{4}$$

Keeping in mind that the attacker has knowledge of the network topology and access to all load measurement. $\boldsymbol{z_a}$ represents the FDI attack, $\boldsymbol{z}$ represents the measurement vector that can be found through power flow analysis between the loads, and $\boldsymbol{a = Hc}$, where $\boldsymbol{H}$ represents the Jacobian matrix that can be found after the power flow measurements passes through state estimation. From the state estimation, the Jacobian matrix is derived, and $\boldsymbol{c}$ represents attack vector, which can be determined based on the dimension of the Jacobian matrix, last but not least $\boldsymbol{e}$ is the error from measurement vector determined by state estimation

5 Alternative Solutions

In regard to the dataset collection, some other solutions have opted to obtaining their measurements from smart meters that they have monitored over a period of time instead of finding a readymade dataset.

5.1 Utilizing an RNN

Qingyu Deng and Jian Sun Beijing Institute of Technology found an FDI countermeasure that utilizes a Recurrent Neural Network (RNN), this networks includes a variety of hidden layers that are make the AI model harder to be fooled by adversaries to inject faulty data. It works with creating loops that are able to persist information.

Figure 12 shows the typical architecture of an RNN [11]:

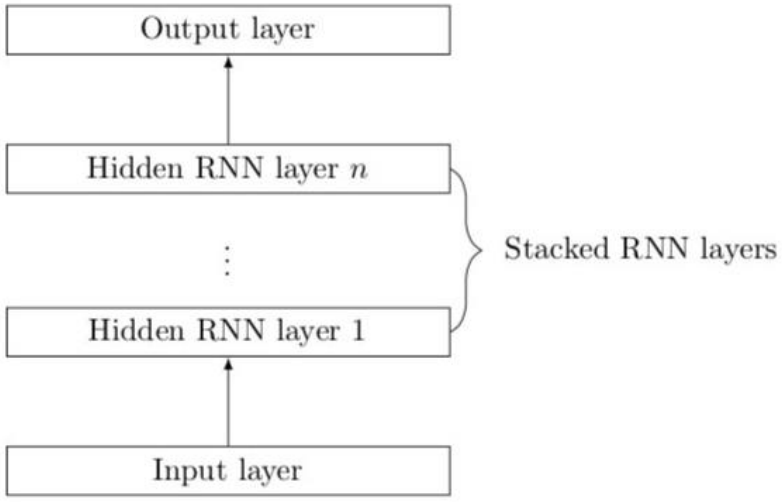


Figure 12 - General Architecture of an RNN

5.2 Utilizing a CNN and an LSTM Network

Xiangyu Niu, Jiangnan Li, Jinyuan Sun, and Kevin Tomsovic have come up with a proposal to detect FDI attacks using an anomaly detector that adopts a Convolutional Neural Network (CNN) along with a Long Short Term Memory (LSTM) network [12]. Those two networks do combined work by estimating system variables, to both observe data measurements and network level features and jointly learn system states.

Figure 13 illustrates the proposed system design of the authors:

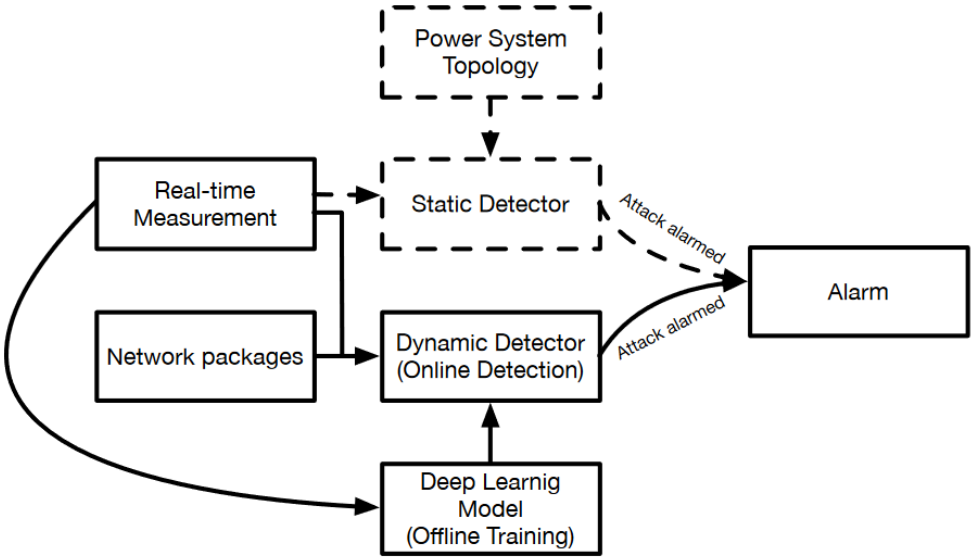


Figure 13 - System Design of Alternative Solution

5.3 Comparison

System Design	Background	Datasets used
Our system design	Depends on our LSTM model using the supervised learning method followed by an MSE comparison to grant maximum accuracy to the system.	Trained and tested using the IHEPCDS dataset with over a million datapoints.
RNN	RNNs are a family of neural networks for processing sequential data, they are also an extension of a convention feedforward neural network. RNN have cyclic connections making them powerful for modeling sequences.	Simulates an IEEE 14-bus system with 11 load buses with the use of the MATPower open-source toolbox in MATLAB.
CNN-LTSM	Consists of a static detector and a deep learning based detection scheme. The static detector can be a State Estimator (SE) or any aforementioned FDI attack detector, which is built independently beyond a dynamic detector. LSTM is a special class of RNN that is designed to avoid the short-term memory problems.	This classifier model was trained utilizing a dataset consisting of daily power consumption data of both normal and fraudulent users in a supervised manner.

Table 2 - Comparison of Our Design with Alternative Solutions

5.4 Conclusion

As seen from Table 2, the RNN-based alternative option is a viable choice to solve the time series forecasting problem, where the future is to be forecasted given historical values. The CNN-LSTM option consists of two parts for its detection system, the first part is where the neural network resides. It is a wise option because the LSTM layers – which is a variant of the RNN layers – has a longer memory and solves many of the problems that come with the vanilla RNN such as the exploding and vanishing gradients. Also, combining it with CNN layers will quicken its training time at the cost of a little bit of performance.

The exploding gradient problem means that the model might unexpectedly fail to converge as it near the convergence point. The vanishing gradient problem causes the training to plateau after a while, preventing the model from improving any further.

To conclude, we will use an LSTM Neural Network (NN) because we are able to forgo the inclusion of the CNN layers in favor of obtaining a higher performance model.

6 Proposed Design

As shown in Figure 3, we have three major components of this system design: power grid parts, attacker parts, and detection system parts. We will go into each one in depth, beginning with

Smart grid component which includes load data from each substation and combined to represents the network-wide dataset. These measurement vectors would be routed to the control center, which is regarded as the core of the power grid system. This center will compile all of the measurements obtained from each substation.

The attacker's side in the design, the attacker will first target the system while the measurement vector is being transmitted to the control center. In the power flow analysis, we will analyze the real power measurements coming from the load data. The measurements will be passed to the state estimator, where the attacker has his own state estimator, which will output a Jacobian matrix, an error vector, and a state vector. Coming to the measurement falsification, which depends on the Jacobian matrix, the error vector, and the measurement vector. The attacker after state estimation will have all the needed variables to perform an FDI attack and will need to create an attack vector that matches the dimensions of the Jacobian matrix. The measurements sent from the substations will be falsified or not, based on a certain percentage of falsification that will be determined. In our design, we first will have not to falsify any measurements at all, and after passing the set of measurements with no falsification, we will have to set a certain percentage of the measurement vector getting falsified. Whether a measurement vector gets falsified or not, we get an output vector which is the Transmitted Measurement Vector. The transmitted measurement vector consists of the real part, or the active powers from the load data in the busses 1 to 14, in addition to power injections, which are a set of only active powers, where reactive powers are neglected.

Detection system part that includes The AI model prediction will forecast the next vector based on historical measurement vectors containing the real power vector and real power flow. The model will receive a certain number of vectors at a given time and will have an outcome based on the historical measurements fed into the AI. Next the predicted vectors are then subjected to MSE estimation, which calculate the differences between the AI predicted vector and the received measurement vector. The calculated MSE is then compared to a pre-selected threshold, and if the MSE exceeds the threshold, the model detects the falsified vector.

6.1 Dataset Collection

We have searched and found many datasets containing measurements from a set of monitored smart meters attached to an active smart grid. These measurements mostly revolved around the power, voltage, and current values of the users' energy consumption, along with their associated timestamps. Many of the datasets we found were limited in size which made us avoid them since the size of the dataset directly affects the accuracy of our generated AI classifier, the larger the dataset and the longer the duration of the smart meter monitoring, the higher the accuracy of our model.

As seen in Table 3, our search for the datasets leads us to the table below from Adrea Dominik, Wilfried, Salvatore and Andrea [13], featuring a group of datasets, the seventh of which was collected in France and has the longest duration of monitoring whereas the fourth which was collected in Italy has the smallest resolution of measurement taking at 1 Hz.

Dataset	Location	Duration	#Houses	#Sensors (per house)	Features	Resolution
ACS-F1 [7]	Switzerland	1 hour session (2 sessions)	N/A	100 devices in total (10 types)	I, V, Q, f, Φ	10 secs
AMPds [8]	Greater Vancouver	1 year	1	19	I, V, pf, F, P, Q, S	1 min
BLUED [5]	Pittsburg, PA	8 days	1	Aggregated	I, V, switch events	12 Khz
GREEND	Austria, Italy	1 year (3-6 months completed)	9	9	P	1 Hz
HES	UK	1 month (255 houses) - 1 year (26 houses)	251	13-51	P	2 min
iAWE [9]	India	73 days	1	33 sensors (10 appliance level)	V, I, f, P, S, E, Φ	1 Hz
IHEPCDS[1]	France	4 years	1	3 circuits	I, V, P, Q	1 min
OCTES[2]	Finland, Iceland, Scotland	4-13 months	33	Aggregated	P, Energy price	7 secs
REDD [4]	Boston, MA	3 - 19 days	6	9-24	Aggregate: V, P; Sub-metered: P	15 Khz (aggr.), 3 sec (sub)
Sample dataset[3]	Austin, TX	7 days	10	12	S	1 min
Smart* [10]	Western Massachussets	3 months	1 Sub-metered +2 (Aggregated + Sub-metered)	25 circuits, 29 appliance monitors	P, S (circuits), P (sub-metered)	1 Hz
Tracebase [6]	Germany	N/A	15	158 devices in total (43 types)	P	1-10 sec
UK-DALE [11]	UK	499 days	4	5 (house 3) - 53 (house 1)	Aggregated P, Sub P, switch-status	16 Khz (aggr.), 6 sec (sub.)

[1] http://tinyurl.com/IHEPCDS
[2] http://octes.oamk.fi/final/
[3] http://www.pecanstreet.org/projects/consortium/

Table 3 - Collection of Datasets Found

The IHEPCDS contains a little more than 2 million datapoints over 4 years. We aim to analyze at least these datasets and use what our analysis shows best. This dataset contains the metrics we need for training, testing, and validation, such as real power, and we can also drive the power flow measurement which will be used to produce FDI attack

In Figure 14, we demonstrate the global active power feature for the first million datapoints from the IHEPCDS dataset [13], where the timeline axis is set to be arbitrary.

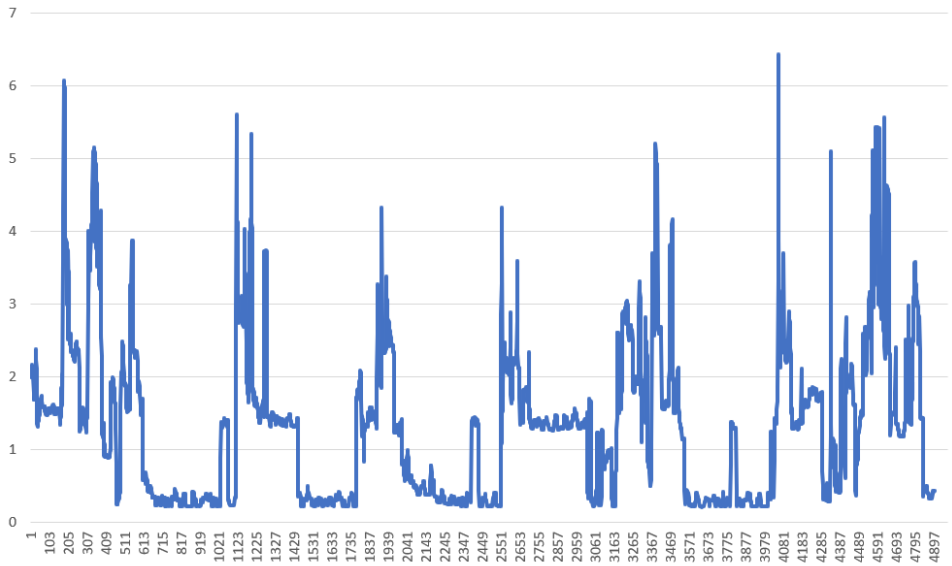


Figure 14 - First Million Datapoints from The IHEPCDS Dataset

6.2 Power System Simulation

As the system design progresses, we decide to build our IEEE bus 14 network system shown in Figure15 using our own power system simulation using MATPOWER. This network is made up of 14 buses. would be taken into account. We will obtain power flow analysis and real power vectors from each substitution of them. These measurements will help to create our own FDIA. Then, using MATLAB, we can simulate the FDIA using the equation (4).

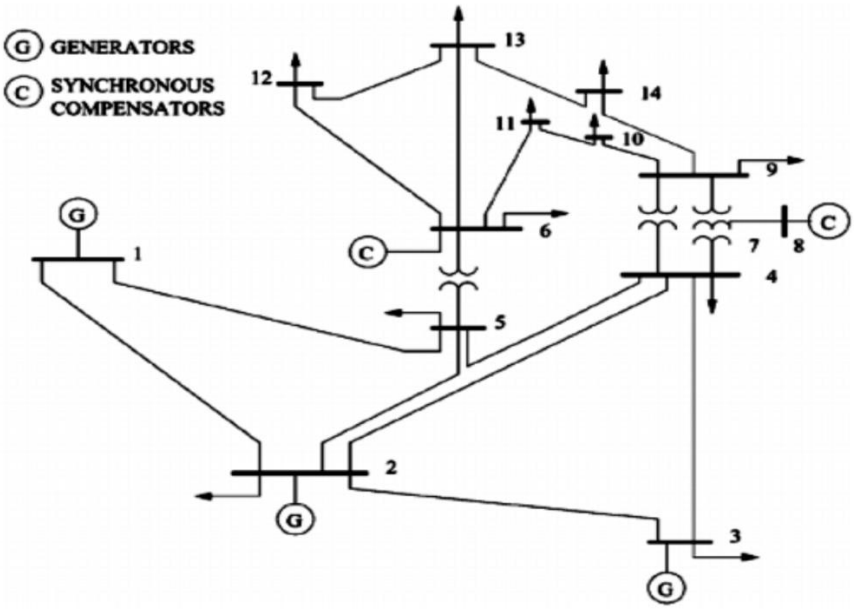


Figure 15 - IEEE 14 Bus Network

6.3 State Estimation

State Estimation gives us the state of the voltage and phase angle for selected dependent measurements from the smart grid. SE can be performed on measurements either from the distribution system or from the customers' smart meters.

The state estimation algorithm is based on the weighted least square method, where the state variables, being the voltage magnitude and phase angle, are calculated from the minimization of the square of the error of all measurements.

A state vector can be estimated using one of three different estimator techniques:

1. Weighted Least Square (WLS) Estimator.
2. Least Absolute Value Estimator.
3. Reweighted Least Squares Estimator (RLS).

We use WLS for our project because WLS and RLS deal with measurements that do not have any electric spikes or any distortion that will affect the sensor. Our dataset is ideal for these estimations. LAV estimation, on the other hand, can cope with distortion, electric spikes, and sensor disabling. We use WLS in particular since it is the most often used in power system simulation.

Figures 16 and 17 show the errors for the voltage magnitudes and the phase angles of different sets of measurements are shown from an experiment conducted by Ahmad and Martin [15] utilizing the WLS estimator method:

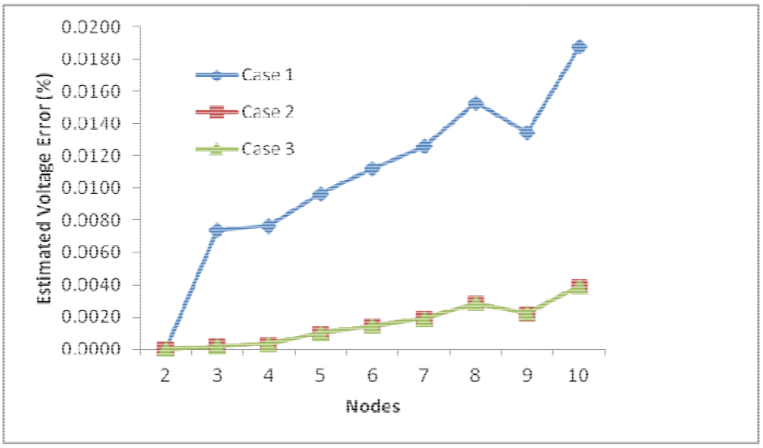


Fig. 3.Estimated Voltage error (%).

Figure 16 - Estimated Voltage Errors

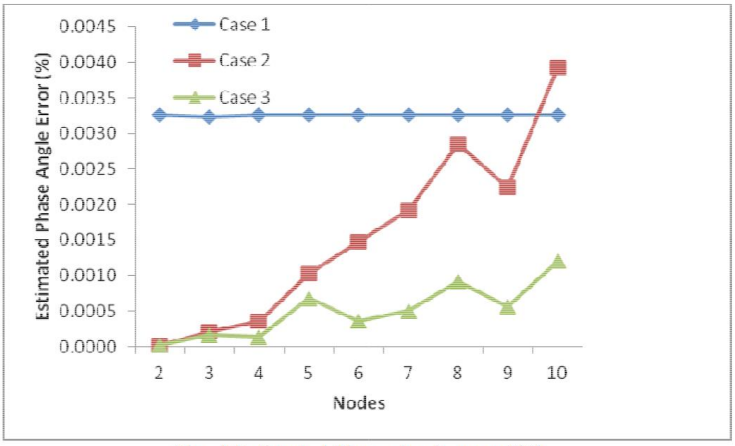


Fig. 4.Estimated Phase Angle error (%).

Figure 17 - Estimated Phase Angle Errors

6.4 Neural Network Structure

As visualized in Figure 18, our Neural Network structure consists of two types of layers, which are dense layers, and LSTM layers. Both layers work to receive the measurement vectors of the first five minutes, and then predicts the measurement vector of the sixth minute. Each layer contains 64 neurons, and those layers output a predicted vector in the sixth minute, which has the size of 25. The justification for incorporating dense layers is to shorten the training time and lower the MAE of the model above the benefits that the LSTM will provide.

As for our activation function, we have used the hyperbolic tangent (tanh) function for our LSTM layers, and the linear function for our dense layers. Using the tanh function enables us to use the CuDNN implementation for our LSTM layers in Keras, Tensorflow. This will utilize the full capabilities of our used local GPU – an RTX 2060, 6 GB VRAM – which would greatly reduce the training time, but more importantly the prediction time. Achieving the lowest possible prediction time is important since our detections system will work in real time.

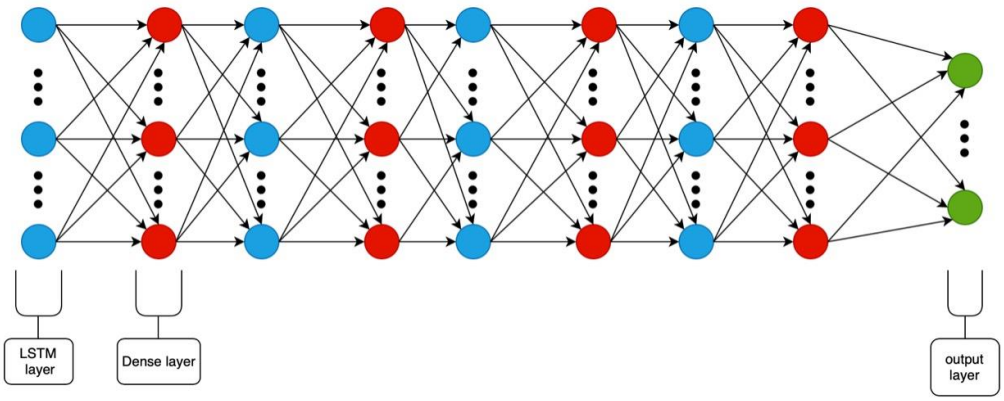


Figure 18 - Neural Network Structure

As per Figure 19, we are planning to take a supervised learning approach to developing our model. And since we are looking to perform fraud detection and classify whether a given set of measurements have had falsified data, we will be building an AI binary classifier. We will write the algorithm using the Python programming language, with the aid of the open-source Tensorflow as our AI framework.

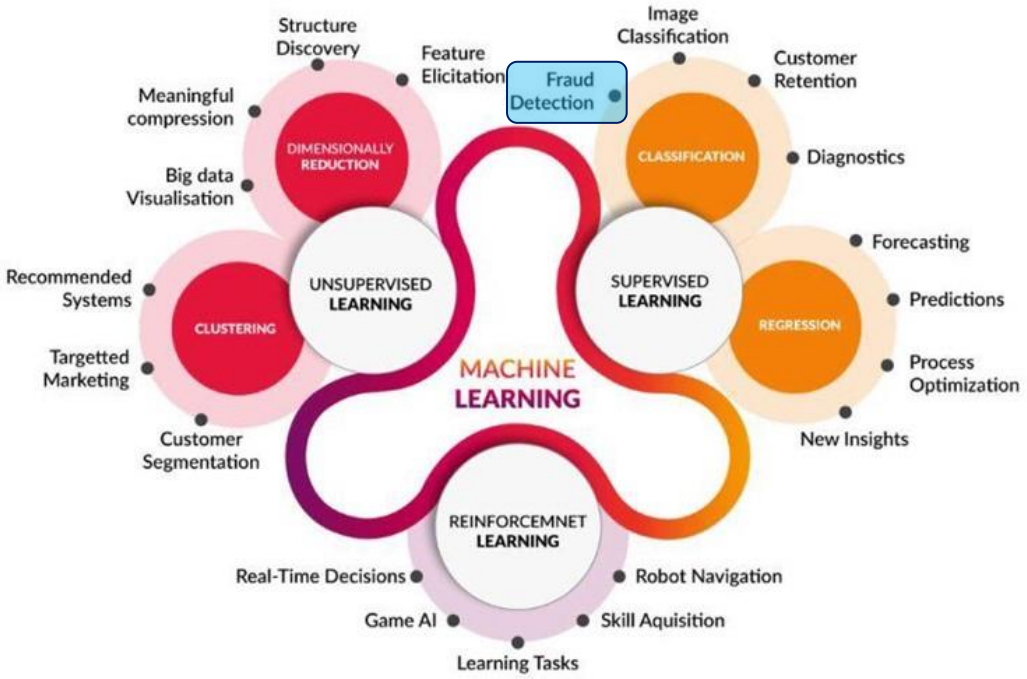


Figure 19 - Uses of Different Types of AI Models

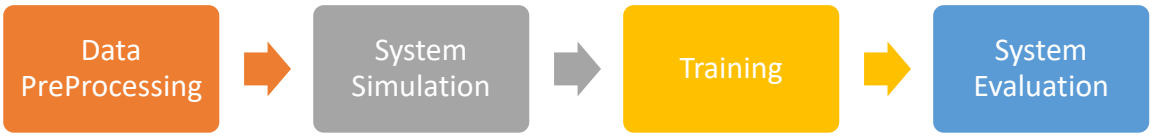
Our systematic procedure towards developing our AI model begins with cleaning the data by removing any entries with missing or obviously faulty values. We will then conduct data analysis on the energy loads for different time periods of the dataset. After extracting potential patterns in the dataset, we will proceed to falsifying randomized measurements in our data while labelling all of the datapoints simultaneously.

The way we plan to simulate FDI and the labelling of our data is by automating it with a Python script. This is possible to due to the simplicity of the process. The falsification of random datapoints with random manipulation percentages within a margin can be done with the use of pseudo-random number generators. Subsequently enabling us to write this script.

Finally, after simulating an FDIA and labelling the data, we can go ahead and use Tensorflow to develop machine learning model to predict the occurrence of FDI within any set of measurements with a certain average confidence percentage.

7 Design Implementation

The proposed countermeasure will be an AI model capable of identifying if an FDI attack has occurred. The following stages will be undertaken to produce the final AI:



Briefly, the four stages of our work stated above begins with the dataset preprocessing, where we rearrange the datasets such that 11 parts symbolize the 11 loads in the network, and keeping the sections close to each other, so instead of two columns, we have 25 columns, 14 of which are real power injections and 11 of which are real power flows. Secondly, system simulation will produce three major output for us that will assist us in progressing to the next stages. Which is the untampered vectors dataset that will be used in the training and tampered vectors with the labels that will be used in the system evaluation, in training phase.70% of the datasets will be used for training, while the remaining 20% will be used for validation and 10% for testing.

Finally, our system evaluation will assess if falsified measurement vectors are observed. Using the mechanism mentioned above, we will evaluate the system's accuracy in detecting FDIAs without any false positives.

7.1 Data Pre-Processing

Before we begin training, our dataset goes through several steps to ensure that it does not produce any invalid results. Let us go through each phase in depth.

The first phase was **cleaning the dataset**, which required us to remove any unwanted strings, such as question marks. Then we must enforce float datatype in active and reactive power to ensure that we are aware of all non-float values.

Following that, the dataset is divided into eleven parts. 3 parts have zero value which are busses number 1,7 and 8. The reason they are zeros is to replicate IEEE's bus14 original behavior. Every substation has two measurements: active power and reactive power. An excerpt of the Python code can be found in appendix 1.1.

As you see in Table 4, after splitting we have to **concatenate** these components side by side. This arrangement will aid us in doing power flow analysis.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	5.36	0.436	0.38	0	0.468	0.21	0.224	0.124	0.304	0.232	0.334	0.098	0.416	0.226	0.306	0.078	0.664	0.07	0.234	0.122	1.668	0.062
2	5.374	0.498	0.382	0	0.402	0.114	0.27	0.17	0.304	0.234	0.332	0.098	0.412	0.224	0.306	0.078	0.264	0	0.234	0.122	1.68	0.062
3	5.388	0.502	0.38	0	0.308	0	0.328	0.258	0.302	0.232	0.332	0.1	0.41	0.226	0.304	0.076	0.264	0	0.232	0.12	1.676	0.062
4	3.666	0.528	0.38	0	0.308	0	0.292	0.206	0.302	0.234	0.33	0.1	0.406	0.224	0.306	0.078	0.264	0	0.232	0.12	1.676	0.06
5	3.52	0.522	0.374	0	0.31	0	0.258	0.156	0.302	0.234	0.328	0.098	0.404	0.222	0.306	0.08	0.26	0	0.232	0.12	1.682	0.06
6	3.702	0.52	0.374	0	0.31	0	0.256	0.154	0.3	0.232	0.326	0.096	0.404	0.226	0.306	0.078	0.258	0	0.378	0.144	1.68	0.06
7	3.7	0.52	0.458	0.126	0.31	0	0.254	0.154	0.298	0.232	0.324	0.096	0.36	0.16	0.306	0.08	0.258	0	0.878	0.21	1.684	0.06
8	3.668	0.51	0.452	0.138	0.312	0	0.254	0.154	0.23	0.134	0.324	0.098	0.332	0.118	0.306	0.08	0.256	0	0.9	0.208	1.676	0.06
9	3.662	0.51	0.45	0.138	0.31	0	0.252	0.154	0.204	0.098	0.324	0.098	0.33	0.118	0.306	0.08	0.256	0	0.91	0.182	1.678	0.06
10	4.448	0.498	0.45	0.14	0.31	0	0.252	0.156	0.202	0.098	0.324	0.098	0.33	0.12	0.306	0.08	0.254	0	0.848	0.082	1.698	0.058

Table 4 – Data Preprocessing Final Results

7.2 System Simulation

The aim of phase is to simulate an IEEE bus 14 system in order to obtain three main things. State estimation, power flow analysis, and FDI attack simulation. These components will talk about it in depth in the following sections. Following that. Our aim is to collect both untampered and tampered vectors and labels that will be used in Training, testing and validation.

7.2.1 Power Flow Analysis

The advantage of power flow analysis is that it takes active and reactive power injections and does DC power flow analysis. This analysis would provide us with the real power flow. These power flows will generate the measurement vectors that we will use later in the FDI simulation. Appendix 2.1 shows an excerpt of our code that executes the power flow analysis.

7.2.2 State Estimation

We used a WLS state estimation in our design; the key aim of the state estimation is to minimize measurement error. Our target in state estimation is to obtain two elements required to launch an attack. The Jacobian matrix and the error vector are these components. As you can see in appendix 2.2, line 75 is where we do the estimation. And we are computing the Jacobian matrix for this network in line 79.

7.2.3 FDI Attack Simulation

As seen recently in FDI attack section [4], we discussed how to simulate an attack with the use of equation (4).

Let us now talk about the process. Appendix 2.3 shows a code excerpt that falsifies the given measurement vector. After the attacker collect the Jacobian matrix and error vector and measurement vector, it is now his turn to create an attack vector based on the dimensions of the Jacobian matrix. For our case it was 25x25 dimension. Following that, all these components will be combined by the attacker to create an attack that will already bypass the Bad Data detector. Table 5 depicts a visual representation of the dataset's falsification distribution.

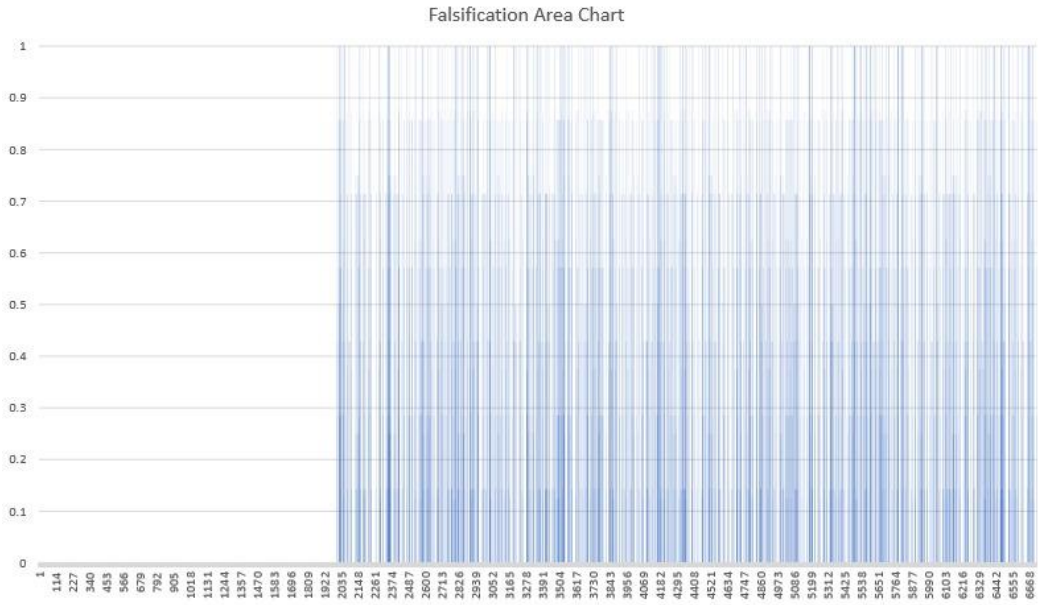
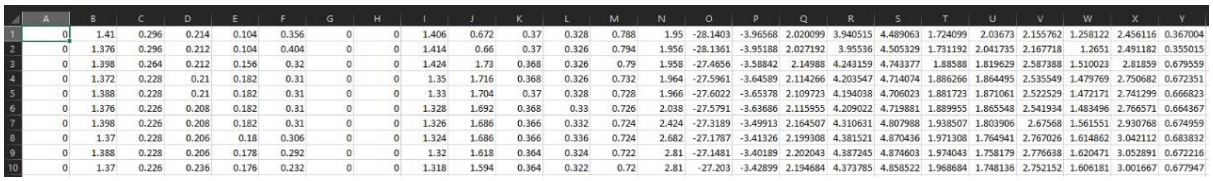


Table 5 - Falsification Distribution

In Table 5, it is the user's duty to feed the detection system 2000 clean vectors to ensure that it works. Then there will be 20 % chance to falsify the dataset.

7.3 AI Training

Figure 20 shows one of the outputs of the system simulation. It shows the first 10 timestamps of the measurements vector dataset. Each timestamp consists of 25 measurements representing the real power injections and flows as discussed earlier.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	0	1.41	0.296	0.214	0.104	0.356	0	0	1.406	0.672	0.37	0.328	0.788	1.95	-28.1403	-3.96568	2.020099	3.940515	4.489063	1.724099	2.03673	2.155762	1.258122	2.456116	0.367004
2	0	1.376	0.296	0.212	0.104	0.404	0	0	1.414	0.66	0.37	0.326	0.794	1.956	-28.1361	-3.95188	2.027192	3.95536	4.505329	1.731192	2.041735	2.167718	1.2651	2.491182	0.355015
3	0	1.398	0.264	0.212	0.156	0.32	0	0	1.424	1.73	0.368	0.326	0.79	1.958	-27.4656	-3.58842	2.14988	4.243159	4.743377	1.88588	1.819629	2.587388	1.510023	2.81859	0.679559
4	0	1.372	0.228	0.21	0.182	0.31	0	0	1.35	1.716	0.368	0.326	0.732	1.964	-27.5961	-3.64589	2.114266	4.203547	4.714074	1.886266	1.864495	2.535549	1.479769	2.750682	0.672351
5	0	1.388	0.228	0.21	0.182	0.31	0	0	1.33	1.704	0.37	0.328	0.728	1.966	-27.6022	-3.65378	2.109723	4.194038	4.706023	1.881723	1.871061	2.522529	1.472171	2.741299	0.666823
6	0	1.376	0.226	0.208	0.182	0.31	0	0	1.328	1.692	0.368	0.33	0.726	2.038	-27.5791	-3.63686	2.115955	4.209022	4.719881	1.889955	1.865548	2.541934	1.483496	2.766571	0.664367
7	0	1.398	0.226	0.208	0.182	0.31	0	0	1.326	1.686	0.366	0.332	0.724	2.424	-27.3189	-3.49913	2.164507	4.310631	4.807988	1.938507	1.803906	2.67568	1.561551	2.930768	0.674959
8	0	1.37	0.228	0.206	0.18	0.306	0	0	1.324	1.686	0.366	0.336	0.724	2.682	-27.1787	-3.41326	2.199308	4.381521	4.870436	1.971308	1.764941	2.767026	1.614862	3.042112	0.683832
9	0	1.388	0.228	0.206	0.178	0.292	0	0	1.32	1.618	0.364	0.324	0.722	2.81	-27.1481	-3.40189	2.202043	4.387245	4.874603	1.974043	1.758179	2.776638	1.620471	3.052891	0.672216
10	0	1.37	0.226	0.236	0.176	0.232	0	0	1.318	1.594	0.364	0.322	0.72	2.81	-27.203	-3.42899	2.194684	4.373785	4.858522	1.968684	1.748136	2.752152	1.606181	3.001667	0.677947

Figure 20 - Measurment Vector Dataset

As shown in the code in appendix 3.1, we are in this part training our AI Model. We, in line 14, use the early stopping callback function, the callback here is used to monitor the validation loss. The callback function will terminate the training if it does not see any improvement in the validation loss within 2 epochs. An epoch is a parameter that defines the number of times our model has completed a training on the whole dataset, meaning that if the dataset is passed forward and backward through the neural network, the number of epochs get incremented by one. As shown in line 19, we use an Adam optimizer, which helps in a faster training and is more popular in research. It is noteworthy to know the importance of the Mean Absolute Error (MAE), which is a training metric that helps us learn about the performance of our model.

Dataset Splitting

When dividing the dataset, we want the model to have as many data points as possible to train on, so the more data points there are for testing, the more accurate the model will be. Also, the more of the dataset we allocate for testing, the more biased the results will be. Therefore, we plan to use 70% of the dataset for training and 20% for validation which is aplenty to obtain a high-performance model. And the remaining 10% will be used to test the model.

Data Windowing

Data windowing is an operation that lets us reshape the hall dataset. Tensorflow would be able to understand it better as a result of this. Following that, the algorithm would take the last five minutes and forecast the next minute depending on these values. Figure 21 will provide a clear description of what we are trying to illustrate.

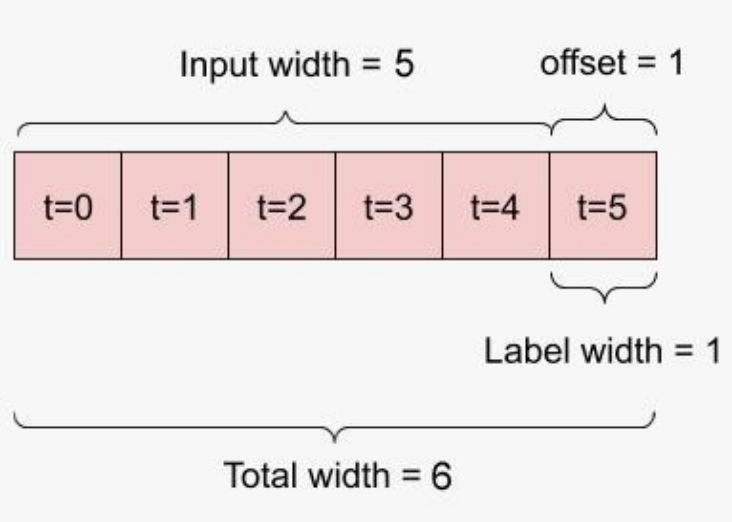


Figure 21- Data Windowing Operation

As shown in the diagram. Four factors influence data windowing.

These factors are **Input width** which will determine how many inputs are going to inject inside the model. Each input considers an array that includes 5 vectors, each vector contains 25 value. Then we have the **offset**. This one decides which minute is predicted. For e.g., suppose we have a 5-minute input width and a 2 offset. The forecast would then be in the 7^{th} minute, with the 6^{th} being ignored. Following that, we have **label width**, which is the model's output (predicted vectors). Finally, we have the **total width**. That is, the hall cycle of injecting and forecasting arrays is essentially the sum of the input width time and the label width time.

Appendix 3.2 shows a code excerpt for the window generating class that we used to create our window with.

Training Results

As we can see in appendix 3.3, these lines are some of our failed attempts at training the model, and after further research and digging deeper in the code, we got the results that are illustrated in Figure 22.

```
Epoch 1/20
2021-04-20 21:34:48.980698: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublas64_11.dll
2021-04-20 21:34:49.952114: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cublasLt64_11.dll
2021-04-20 21:34:50.814608: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library cudnn64_8.dll
1470/1470 [==============================] - 33s 16ms/step - loss: 1.8175 - mean_absolute_error: 0.8196 - val_loss: 0.9566 - val_mean_absolute_error: 0.6686
Epoch 2/20
1470/1470 [==============================] - 22s 15ms/step - loss: 0.7462 - mean_absolute_error: 0.5892 - val_loss: 0.7516 - val_mean_absolute_error: 0.5874
Epoch 3/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.6875 - mean_absolute_error: 0.5566 - val_loss: 0.7181 - val_mean_absolute_error: 0.5671
Epoch 4/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.6589 - mean_absolute_error: 0.5431 - val_loss: 0.6959 - val_mean_absolute_error: 0.5598
Epoch 5/20
1470/1470 [==============================] - 22s 15ms/step - loss: 0.6522 - mean_absolute_error: 0.5415 - val_loss: 0.7226 - val_mean_absolute_error: 0.5646
Epoch 6/20
1470/1470 [==============================] - 22s 15ms/step - loss: 0.6120 - mean_absolute_error: 0.5234 - val_loss: 0.6696 - val_mean_absolute_error: 0.5497
Epoch 7/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.5778 - mean_absolute_error: 0.5041 - val_loss: 0.6815 - val_mean_absolute_error: 0.5532
Epoch 8/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.5708 - mean_absolute_error: 0.5017 - val_loss: 0.6339 - val_mean_absolute_error: 0.5275
Epoch 9/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.5639 - mean_absolute_error: 0.4969 - val_loss: 0.6317 - val_mean_absolute_error: 0.5282
Epoch 10/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.5637 - mean_absolute_error: 0.4976 - val_loss: 0.6347 - val_mean_absolute_error: 0.5329
Epoch 11/20
1470/1470 [==============================] - 21s 14ms/step - loss: 0.5682 - mean_absolute_error: 0.5006 - val_loss: 0.6325 - val_mean_absolute_error: 0.5279

Process finished with exit code 0
```

Figure 22 - Training Results

As seen in Figure 22, we can see the results of the validation losses and the mean absolute errors. We can notice that we've reached to 0.5006 in the mean absolute error, and 0.6325 in the validation loss. As seen in the last two epochs, the validation loss did not improve, and in result, the early stopping callback function stopped the training although the training was not complete to the total number of epochs which is 20. We can notice the training time, where it took the model 256 seconds, or 4 minutes and 6 seconds, to train 11 epochs.

7.4 System Evaluation

In this section of the project. The system evaluation would assess the forecast vector that was sent from the model by calculating the current threshold between the predicted and obtained measurement vectors. If the current threshold exceeds the pre-selected threshold, an FDI attack is observed. More details of the evaluation will be discussed in 7.4.1 and 7.4.2.

Comparing Different Threshold Metrics

In this section, we are trying to find the perfect threshold metric that would suit our case, where we want to differentiate the tampered timestamps from the untampered ones, or the cleaned samples from the falsified samples. Below, we are going to compare three different metrics and explain why we chose one. Our goal in this comparison is to find a way to differentiate between the first 2000 measurements and the rest of the readings.

As shown in Figure 23, we can see that there is not any type of differentiation in the measurements while using the mean absolute error, and all measurements seem to be random.

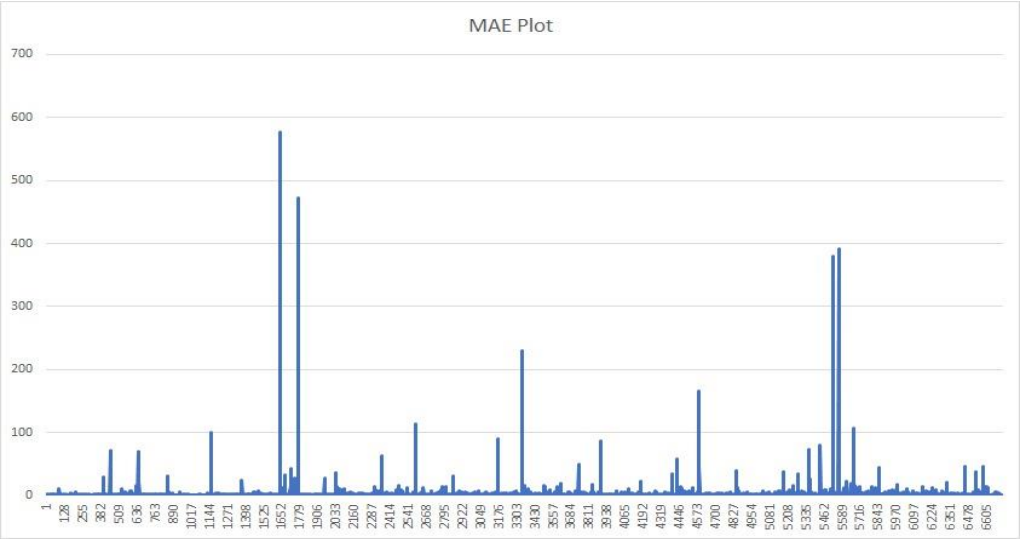


Figure 23 - Mean Absolute Error Plot

We then tried the mean absolute error squared, which is squaring the mean absolute error, in order to get somehow a stable plot after squaring, but noticed, as seen in Figure 24, that squaring small values makes them nearly vanish, and squaring large values makes them even larger, which was not suitable to our case.

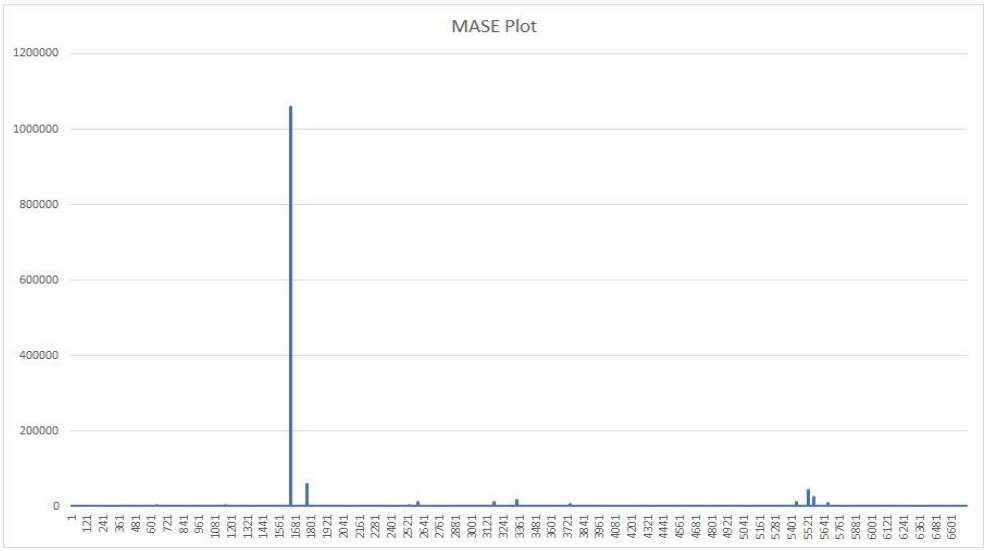


Figure 24 - Mean Absolute Squared Error Plot

At last, we tried the mean square error, which seemed to be the most suitable one among the different metrics. It is noticeable in Figure 25 that there is a visible differentiation between first 2000 measurements, which are nearly 0, and the rest which are the falsified readings. Given the scale of smart grid system and the loads, we found the best MSE threshold to be 20.

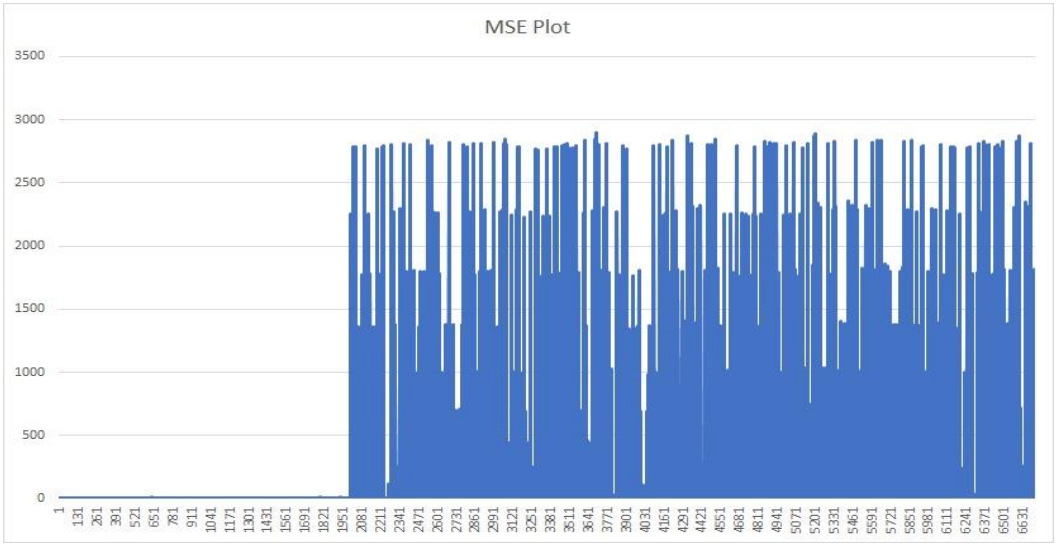


Figure 25 - Mean Square Error Plot

Evaluation Results

Regarding the Evaluation results, an excerpt can of the evaluation code can be found in the appendix 4.1. As you can see, we used the AI model's prediction to provide us with forecasting to compare. The MSE then calculates the current threshold for each iteration from lines 51 to 54. Following that, from lines 56 to 66, the algorithm would compare to see if the current threshold is greater than 20, if so, there is an FDI attack, and it will be counted in the number of success if it is a match the corresponding label. There will be no FDI if the threshold is not exceeded but also it will be counted in the number of success if the labels are matching the prediction. This way, our measure of accuracy will include both true positives and true negatives. Then we can calculate the accuracy (success rate) by dividing all succeeded predictions by the number of total predictions. Our success rate reached up to **100%**.

In equation (4), a is a vector that is large in size, the difference between Za and z will be large for some but not all values in the vector since not all differences are large depending on the vector **a**, the attack will bypass the conventional Bad Data Detector, however it will not bypass our MSE-based detection system, because it does not include any division by measurements. This is why as we have proved; it is possible and reasonable to detect 100% of the Fault Data Injection attacks.

8 Conclusion and Future Work

In conclusion, we have discussed and implemented in this report an FDI Detection system that detects the BDD-bypassing FDIA, which is the most sophisticated and dangerous attack on any smart grid. We have simulated the smart grid operation using MATPOWER and the FDIA itself using MATLAB, before implementing our detection system in Python. We have trained an LSTM NN for our predictor and used MSE as our threshold comparison metric.

We have obtained a 100% detection accuracy from our real-time detection system and have proven that it is practically possible and reasonable to do so.

Future work could include coming up with a proposal that is not problem-dependent, but more importantly does not require any initially fed clean measurements, this is because in the real world, we can never be completely sure of any set of measurements to be fully clean to initially feed the predictor.

9 List of Constraints

The following constraints are applied to our specific system design:

1. In the case that the input set of measurements in small in size, the output may not be accurate sine the AI model will be trained using relatively larger datasets.
2. The MSE threshold is problem dependent. It therefore must be adjusted for each particular smart grid the detection system is deployed in.
3. Environment variables such as the efficiency of the transformers where the training datasets have been developed may cause environment-specific biases in the output of the system.

9.1 Environmental Concerns

Our system design will not impose any negative effects on the environment, in fact, detecting FDI attacks in order to deal with them would cause less stolen energy, reducing the types of energy production that are harmful to the environment and encourages renewables energy technologies.

9.2 Ethical and Social Issues

The UAE, especially Dubai, is aggressively moving towards renewable energy technologies and building smart cities, and smart grids is one of the most important elements in building a smart city dependent on renewable energy sources.

Our contribution helps ensure the security of the operations of smart grids, making them more robust and reliable, hence more viable to be put into practice more fundamentally in the country.

9.3 Economic Issues

Our project is a fully software-based system which will make use of datasets already present amongst the research community, therefore we will have no financial spending on it.

Other products in the market may compete with our proposed system in case they perform a more generic and load analysis before attempting to train their AI models. On the other hand, we have leveraged the use of state estimation to aid the decision making of our system, which will act as a unique selling point and give us a competitive edge.

10 List of Specifications

10.1 Parts List and Estimated Budget

Our AI classifier would need GPU processing power to train, which makes us need access to the Saqr GPU cluster in the university.

Since our project is fully done in software with no proprietary software used, we need no further resources.

10.2 Timeline

Figure 26 - Senior Design Project 2 Timeline

References

[1] Vijayapriya, Tamilmaran. (2011). Smart Grid: An Overview. Smart Grid and Renewable Energy. 02. 305-311. 10.4236/sgre.2011.24035.

[2] Deka, Deepjyoti & Baldick, R. & Vishwanath, Sriram. (2015). Jamming Aided Generalized Data Attacks: Exposing Vulnerabilities in Secure Estimation. 10.1109/HICSS.2016.319.

[3] D. Singh, R.K. Misra, V.K. Singh, R.K. Pandey. (2010). Bad data pre-filter for state estimation, International Journal of Electrical Power & Energy Systems, 32, 1165-1174.

[4] D. Wang, X. Guan, T. Liu, Y. Gu, Y. Sun and Y. Liu, "A survey on bad data injection attack in smart grid," 2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), Kowloon, 2013, pp. 1-6, doi: 10.1109/APPEEC.2013.6837157.

[5] McCary, Eric & Xiao, Yang. (2015). Smart Grid Attacks and Countermeasures. EAI Endorsed Transactions on Industrial Networks and Intelligent Systems. 2. 10.4108/inis.2.2.e4.

[6] Souhila Aoufi, Abdelouahid Derhab, Mohamed Guerroumi. (2020). Survey of false data injection in smart power grid: Attacks, countermeasures and challenges, Journal of Information Security and Applications, 54.

[7] Ahmed, M., Pathan, AS.K. False data injection attack (FDIA): an overview and new metrics for fair evaluation of its countermeasure. Complex Adapt Syst Model 8, 4 (2020). https://doi.org/10.1186/s40294-020-00070-w

[8] Jacob Sakhnini, Hadis Karimipour, Ali Dehghantanha, Reza M. Parizi, Gautam Srivastava. (2019). Security aspects of Internet of Things aided smart grids: A bibliometric survey, Internet of Things.

[9] W. Li, T. Logenthiran, V. Phan and W. L. Woo, "A Novel Smart Energy Theft System (SETS) for IoT-Based Smart Home," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5531-5539, June 2019, doi: 10.1109/JIOT.2019.2903281.

[10] R. Jiang, R. Lu, Y. Wang, J. Luo, C. Shen and X. Shen, "Energy-theft detection issues for advanced metering infrastructure in smart grid," in Tsinghua Science and Technology, vol. 19, no. 2, pp. 105-120, April 2014, doi: 10.1109/TST.2014.6787363.

[11] Q. Deng and J. Sun, "False Data Injection Attack Detection in a Power Grid Using RNN," IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, 2018, pp. 5983-5988, doi: 10.1109/IECON.2018.8591079.

[12] X. Niu, J. Li, J. Sun and K. Tomsovic, "Dynamic Detection of False Data Injection Attack in Smart Grid using Deep Learning," 2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Washington, DC, USA, 2019, pp. 1-6, doi: 10.1109/ISGT.2019.8791598.

[13] A. Monacchi, D. Egarter, W. Elmenreich, S. D'Alessandro and A. M. Tonello, "GREEND: An energy consumption dataset of households in Italy and Austria," 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, 2014, pp. 511-516, doi: 10.1109/SmartGridComm.2014.7007698.

[14] Mostafa Mohammadpourfard, Ashkan Sami, Ali Reza Seifi. (2017). A statistical unsupervised method against false data injection attacks: A visualization-based approach, Expert Systems with Applications, 84, 242-261.

[15] A. Abdel-Majeed and M. Braun, "Low voltage system state estimation using smart meters," 2012 47th International Universities Power Engineering Conference (UPEC), London, 2012, pp. 1-6, doi: 10.1109/UPEC.2012.6398598.

Appendices

Below is a reference list of all the items present in the appendices and their corresponding page numbers.

1 Data Preprocessing 45

1.1 Code Excerpt 45

2 System Simulation 46

2.1 Power Flow Analysis Code Excerpt 46

2.2 State Estimation Code Excerpt 46

2.3 FDIA Simulation Code Excerpt 46

3 Training 47

3.1 Training Function Code 47

3.2 Data Window Generator Class Code Excerpt 47

3.3 Some Failed Training Attempts 48

4 System Evaluation 49

4.1 Code Excerpt 49

1 Data Preprocessing

1.1 Code Excerpt

```
# Split the dataframe into eleven
pieceSize = int(len(df) / 11)
dfs = []
for i in range(0, 11):
    startingIndex = (pieceSize * i) + 1  # Will ignore the first row
    endingIndex = pieceSize * (1 + i)
    splitDf = df[startingIndex:endingIndex].reset_index(drop=True)  # Reindex

    # Rename columns
    activeName = 'Active' + str(i)
    reactiveName = 'Reactive' + str(i)
    splitDf.columns = [activeName, reactiveName]

    dfs.append(splitDf)

# Merge the two dataframes side by side
df = pd.concat(dfs, axis=1)
```

2 System Simulation

2.1 Power Flow Analysis Code Excerpt

```
%-----------------------------------------%
% Set the case file parameters
cell = 1;
for i = 2:14
    if (i ~= 7) && (i ~= 8)
        mpc.bus(i,2) = 1; % Set as load bus
        mpc.bus(i,3) = actives(index,cell);
        mpc.bus(i,4) = reactives(index,cell);
        cell = cell + 1;
    end
end
% Set as generator buses
mpc.bus(7,2) = 2;
mpc.bus(8,2) = 2;

% Save the casefile and run the power flow analysis
savecase('CustomCase14.m', mpc);
results = rundcpf(mpc);

%-----------------------------------------%
% Create the measurement vector
realPowerInjections = results.bus(:,3);
realPowerFlows = results.branch(:,14);

measurementVector = [realPowerInjections; realPowerFlows];
```

2.2 State Estimation Code Excerpt

```
%-----------------------------------------%
% Perform state estimation
ser = StateEstimation(results);

%-----------------------------------------%
% Compute the Jacobian matrix for the bus system
H = makeJac(mpc);

jacLen = length(H);

%-----------------------------------------%
% Save measurement vector to untamperedVectors
measurementVector = measurementVector(1:jacLen);
untamperedVectors = [untamperedVectors, measurementVector];
```

2.3 FDIA Simulation Code Excerpt

```
        % Compute error vector
        error = [];
        for i = 1:14
            error = [error; 0];
        end

        ser.z = ser.z(1:20);
        ser.z_est = ser.z_est(1:20);
        flowsError = ser.z - ser.z_est;
        error = [error; flowsError];
        error = error(1:jacLen);

        % Compute attack vector
        r = randi([1 10], 1, 1); % Generate a random number from 0 to 100

        c = [];
        for i = 1:jacLen
            c = [c, r];
        end
        c = transpose(c);

        % Compute the falsified measurement vector
        za = measurementVector + (H*c) + error;
```

3 Training

3.1 Training Function Code

```python
def compileAndFit(inputModel, inputWindow, patience=2):
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                      patience=patience,
                                                      mode='min')

    inputModel.compile(loss=tf.losses.MeanSquaredError(),
                       optimizer=tf.optimizers.Adam(learning_rate=0.01),
                       metrics=[tf.metrics.MeanAbsoluteError()])

    inputModel.fit(inputWindow.train, epochs=20,
                   validation_data=inputWindow.val,
                   callbacks=[early_stopping])

    inputModel.save(r'Model\model.h5')

    return inputModel
```

3.2 Data Window Generator Class Code Excerpt

```python
class WindowGenerator:
    def __init__(self, input_width, label_width, shift,
                 train_df, val_df, test_df,
                 label_columns=None):
        # Store the raw data.
        self.train_df = train_df
        self.val_df = val_df
        self.test_df = test_df

        # Work out the label column indices.
        self.label_columns = label_columns
        if label_columns is not None:
            self.label_columns_indices = {name: i for i, name in
                                          enumerate(label_columns)}
        self.column_indices = {name: i for i, name in
                               enumerate(train_df.columns)}

        # Work out the window parameters.
        self.input_width = input_width
        self.label_width = label_width
        self.shift = shift

        self.total_window_size = input_width + shift

        self.input_slice = slice(0, input_width)
        self.input_indices = np.arange(self.total_window_size)[self.input_slice]

        self.label_start = self.total_window_size - self.label_width
        self.labels_slice = slice(self.label_start, None)
        self.label_indices = np.arange(self.total_window_size)[self.labels_slice]
```

3.3 Some Failed Training Attempts

IHEPCDS Dataset Training - 60 Hz (original)													
Attempt	Dataset division	Total # layers	Max(neurons/L)	Layers' types	Activation function(s)	Learning rate	Decay rate	# of Epochs	Loss	MAE	Loss_Val	MAE_Val	Training time
1	90 - 5 - 5	4	256	CNN, DNN, Dropout	relu, linear	0.01	2.50E-04	20	0.0735	0.1302	0.045	0.0753	~16 minutes
2	90 - 5 - 5	4	1024	CNN, DNN, Dropout	relu, linear	0.01	5.00E-04	20	0.0666	0.1154	0.0438	0.0767	~120 minutes
3	90 - 5 - 5	5	1024	CNN, DNN, Dropout	relu, linear	0.01	5.00E-04	20	0.0682	0.1197	0.0448	0.0769	~170 minutes
4	90 - 5 - 5	3	1024	CNN, DNN, Dropout	relu, linear	0.01	5.00E-04	20	0.0661	0.1138	0.0444	0.0787	~65 minutes
5	90 - 5 - 5	3	1024	CNN, DNN, Dropout	relu, linear	0.001	5.00E-05	20	0.0624	0.105	0.0431	0.0794	~68 minutes
6	90 - 5 - 5	3	1024	CNN	relu, linear	0.001	5.00E-05	20	0.0565	0.0886	0.0413	0.0413	~61 minutes
7	90 - 5 - 5	2	1024	CNN	relu, linear	0.001	5.00E-05	20	0.0566	0.088	0.0414	0.0724	~8 minutes
8	90 - 5 - 5	2	1024	CNN	relu, linear	0.001	2.50E-05	40	0.056	0.0875	0.041	0.0721	~16 minutes
9	90 - 5 - 5	2	2048	CNN	relu, linear	0.001	5.00E-05	20	0.0566	0.0879	0.0414	0.0733	~10 minutes
10	90 - 5 - 5	4	32	CNN, LSTM	relu, linear	0.001	5.00E-05	20	0.0569	0.0882	0.0415	0.0743	~36 minutes
11	90 - 5 - 5	7	32	CNN	relu	0.001	3.33E-06	300	0.0518	0.0824	0.041	0.0712	~198 minutes
12	90 - 5 - 5	7	128	CNN	relu	0.001	3.33E-06	300	0.0457	0.0774	0.0446	0.0745	~252 minutes
IHEPCDS Dataset Training - 300 Hz													
Trainin no.	Dataset division	Total # layers	Max(neurons/L)	Layers' types	Activation function(s)	Learning rate	Decay rate	# of Epochs	Loss	MAE	Loss_Val	MAE_Val	Training time
13	90 - 5 - 5	7	128	CNN	relu	0.001	3.33E-06	300	0.1067	0.1625	0.1273	0.174	~53 minutes
IHEPCDS Dataset Training - 900 Hz													
Trainin no.	Dataset division	Total # layers	Max(neurons/L)	Layers' types	Activation function(s)	Learning rate	Decay rate	# of Epochs	Loss	MAE	Loss_Val	MAE_Val	Training time
14	90 - 5 - 5	7	512	CNN	relu	0.001	6.66E-06	150	0.1835	0.2475	0.3042	0.3359	~43 minutes
IHEPCDS Dataset Training - 60 Hz (original)													
Trainin no.	Dataset division	Total # layers	Max(neurons/L)	Layers' types	Activation function(s)	Learning rate	Decay rate	# of Epochs	Loss	MAE	Loss_Val	MAE_Val	Training time
15	70 - 15 - 15	4	64	LSTM	tanh	0.001	5.00E-05	20	0.2624	0.1829	0.0924	0.1025	~79 minutes
16	70 - 15 - 15	7	32	CNN (3), LSTM (3)	tanh, linear	0.001	5.00E-05	20	0.0586	0.0889	0.0418	0.0793	~43 minutes
17	70 - 15 - 15	5	256	CNN (2), LSTM (2)	tanh, linear	0.01	3.33E-05	7	0.0687	0.1199	0.0469	0.1026	~58 minutes
18	70 - 15 - 15	3	256	CNN (1), LSTM (1)	tanh, linear	0.01	3.33E-05	32	0.0808	0.1065	0.0572	0.0889	~118 minutes
19	70 - 15 - 15	7	64	CNN (3), LSTM (3)	tanh, linear	0.001	3.33E-06	21	0.0777	0.1028	0.057	0.0905	~60 minutes
20	70 - 15 - 15	7	64	LSTM	tanh, linear	0.001	3.33E-06	26	0.0795	0.1041	0.0569	0.0882	~112 minutes
21	70 - 15 - 15	7	64	CNN	tanh, linear	0.001	3.33E-06	13	0.1679	0.1681	0.1174	0.1406	~10 minutes
22	70 - 15 - 15	7	512	CNN	tanh, linear	0.001	3.33E-06	17	1.3787	0.9016	0.8913	0.8117	~100 minutes
23	70 - 15 - 15	7	256	LSTM	tanh, linear	0.001	3.33E-06	30	0.0791	0.1038	0.0565	0.0889	stopped@ ~10 hrs
24	70 - 15 - 15	7	128	LSTM	tanh, linear	0.001	3.33E-06	7	0.0626	0.097	0.0507	0.0917	stopped@ ~10 hrs
25	70 - 15 - 15	7	32	CNN (3), LSTM (3)	tanh, linear	0.001	3.33E-06	32	0.0558	0.0892	0.0511	0.0888	~5 hrs 17 mins
26	70 - 15 - 15	7	32	CNN (3), LSTM (3)	tanh, linear	0.001	3.33E-06	56	0.0643	0.0954	0.0501	0.0863	~7 hrs 18 mins
27	70 - 15 - 15	2	64	LSTM	sigmoid, linear	0.01	3.33E-05	192	0.0827	0.1057	0.0559	0.0865	~8 hrs 14 mins

4 System Evaluation

4.1 Code Excerpt

```
# Predictions
predictions = model.predict(window.test)
extractedPredictions = []
for i in range(len(predictions)):
    extractedPredictions.append(predictions[i][4])


numOfSuccesses = 0
mseList = []
for i in range(len(extractedPredictions)):
    currentMse = mse(tamperedList[i], extractedPredictions[i])
    mseList.append(currentMse)

    if currentMse > 20:
        detected = True
    else:
        detected = False

    success = labelsList[i] == detected
    if success:
        numOfSuccesses += 1

    if i != 0 and ~success:
        print(labelsList[i], currentMse)

successRate = (numOfSuccesses/len(extractedPredictions)) * 100
```