# PROJECT REPORT

## Graph-Based Intrusion Detection in Cyber-Physical Systems

submitted to the
Marshall Plan Scholar Foundation

submitted by:

**Stefan Binna, BSc**

Marshallplan-Jubiläumsstiftung
Austrian Marshall Plan Foundation
Fostering Transatlantic Excellence

FH Salzburg

USC University of Southern California

Supervisor: FH-Prof. Priv.-Doz. DI Mag. Dr. Dominik Engel

Supervisor: Prof. Viktor K. Prasanna

Los Angeles & Salzburg, September 2018

# General Information


| | |
|---|---|
| First Name, Last Name: | Stefan Binna, BSc |
| Keywords: | Intrusion Detection, Cyber-Physical Systems, Machine Learning, State Estimation, Smart Grid |
| Academic Supervisor: | FH-Prof. Priv.-Doz. DI Mag. Dr. Dominik Engel |
| Academic Supervisor: | Prof. Viktor K. Prasanna |


# Abstract


Intelligent energy networks, a special type of cyber-physical systems, rely on state estimation in order to determine whether the power grid is operating properly, or not. An invalid state estimate can have a huge impact on the stability of the grid and can cause severe socioeconomic damage. False Data Injection Attacks (FDIAs) display a prominent threat to the operation of power systems, especially when carefully constructed to bypass the traditional Bad Data Detector (BDD). Therefore, an Intrusion Detection System (IDS) has to be in place to prevent FDIAs from going unnoticed. A major limitation of current approaches is that only coarse-grained attack detection is performed. In order to take effective mitigation actions, it would be more beneficial to detect whether any critical subset of state variables is under attack or not. In this thesis, two state-of-the-art machine learning algorithms are investigated for subset level detection of FDIAs. Furthermore, the trade-off between performance and subset size is investigated. The proposed detection algorithms are evaluated by simulating FDIAs on the IEEE 30-bus system using real-world load data for measurement construction.

# Table of Contents

# List of Abbreviations

| | |
|---|---|
| **AC** | Alternating Current |
| **Acc** | Accuracy |
| **AD** | Anomaly Detection |
| **AMI** | Advanced Metering Infrastructure |
| **ANN** | Artificial Neural Network |
| **AUC** | Area Under Curve |
| **BDD** | Bad Data Detector |
| **BP** | Backpropagation |
| **BPTT** | Backpropagation Through Time |
| **CCPA** | Coordinated Cyber-Physical Attack |
| **CDBN** | Conditional Deep Belief Network |
| **CEC** | Constant Error Carousel |
| **DBN** | Deep Belief Network |
| **DC** | Direct Current |
| **DRE** | Density Ratio Estimation |
| **ENN** | Extended Nearest Neighbor |
| **FDIA** | False Data Injection Attack |
| **GLR** | Generalized Likelihood Ratio |
| **GLRT** | Generalized Likelihood Ratio Test |
| **GPS** | Global Positioning System |
| **GRU** | Gated Recurrent Unit |
| **HTI** | Hypothesis Testing Identification |
| **IDS** | Intrusion Detection System |
| **IT** | Information Technology |
| **IPS** | Intrusion Prevention System |
| **ISE** | Interval State Estimation |
| **KLD** | Kullback-Leibler Distance |
| **kNN** | k-nearest Neighbor |

**LMP** Locational Marginal Price
**LR** Load Redistribution
**LSTM** Long Short-Term Memory
**MAPE** Mean Absolute Percentage Error
**MLP** Multi-Layer Perceptron
**MSA** Margin Setting Algorithm
**NIST** National Institute of Standards and Technology
**NN** Neural Network
**NYISO** New York Independent System Operator
**OPF** Optimal Power Flow
**PCA** Principal Component Analysis
**PCS** Process Control System
**PDC** Phasor Data Concentrator
**PMU** Phasor Measurement Unit
**Prec** Precision
**QD** Quickest Detection
**Rec** Recall
**RNN** Recurrent Neural Network
**ROC** Receiver Operating Characteristic
**RTU** Remote Terminal Unit
**S3VM** Semisupervised SVM
**SAE** Stacked Auto-Encoder
**SCADA** Supervisory Control and Data Acquisition
**SCADA/EMS** Supervisory Control and Data Acquisition/Energy Management System
**SCED** Security-Constrained Economic Dispatch
**SLR** Sparse Logistic Regression
**SVE** State Vector Estimation
**SVM** Support Vector Machine
**WLS** Weighted Least Squares
**WWW** World-Wide Web

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The area of cyber-physical systems has evolved drastically in recent years, leading to a massive increase in complexity [1]. According to Lee [2], cyber-physical systems have the potential to revolutionize Information Technology (IT) development over the next few decades, entirely changing the way how humans physically interact with the digital world.

One special type of cyber-physical systems are intelligent energy networks, also known as *smart grids*. A smart grid consists of an Advanced Metering Infrastructure (AMI), multiple power plants and substations, as well as communication networks to communicate sensor reading, control information and distributed power generation. In contrast to the traditional power grid, where the main purpose is to transfer energy from a few central generators to a varying number of customers, the smart grid extends this functionality by allowing a two-way flow of both electricity and information [3].

The upgrade from a traditional power grid to a smart grid requires careful consideration of secure infrastructures and architectures [4]. Whenever new and highly sophisticated systems of large capacity are introduced, vulnerabilities have to be taken into consideration and must not be neglected. Thus, cyber-physical threats will be present and have to be dealt with.

In a smart grid, all data from the communication network is processed at Supervisory Control and Data Acquisition/Energy Management Systems (SCADA/EMSs) to continuously monitor the operating conditions of the grid. One major functionality of a smart grid is *state estimation*, which is used to convert meter measurements into a single system state using power flow models [5]. The resulting state estimate is then used as a basis for various tasks, such as topology processing, electricity pricing and

load shedding. It is thus needless to say, that the correctness of the state estimate is of major importance for the stability of a smart grid.

## 1.1 Problem Statement and Objectives

As already mentioned, the correctness of the state estimate is utterly important for the operation of a smart grid. In order to protect the state estimation against attacks, a secure architecture must be deployed implementing concepts for prevention, detection and resilience [6].

In recent years, several works have been published, proposing methods for both detection of attacks, protection against attacks and creation of different types of attacks. The problem with only implementing a protection-based approach is, that protection can never be 100% secure at all times and therefore attacks may bypass the implemented protection scheme. In order to mitigate this possibility, the implementation of a second layer of defense featuring a detection-based approach, is important.

The recent increase in the popularity of machine learning led to various works considering different machine learning-based algorithms and settings in order to detect attacks. A major limitation of the current approaches is, that detection methods only classify whether a grid is under attack or not. In order to take effective mitigation actions, it would be more beneficial to detect whether any of the critical state estimation variables are under attack or not. Moreover, many works hardly describe the details of the generation of attacks, making it difficult to reproduce and verify the results. The same applies to the documentation of simulations and machine learning models. Although there already exist a few surveys on defense methods against attacks on state estimation, to the best of the author's knowledge no survey summarizing the large amount of research works up to the time of writing this thesis, is available.

The need for a solution to these problems leads to the objectives of this thesis, which are defined below.

- **Extensive literature survey.** A literature survey, investigating all different types of attacks and defense methods, will be given. The objective of the literature survey is to break down the large amount of research works available in the area of attacks on state estimation and its corresponding defense methods in the power grid into clearly defined categories. Additionally, all latest works up to the time of writing will be considered.

- **Test framework.** A clear and concise test framework is needed in order to generate reproducible data sets and clearly document simulations. One major component of the test framework will be the aforementioned data generator, which will allow the generation of both measurement data as well as attacks, featuring different attack strategies and energies. The test framework will be implemented according to the concepts of object-oriented programming, making it easily maintainable and extendable.
- **Subset level.** An algorithm will be presented that allows the construction of attacks on a subset level by partitioning the state variables into the required number of subsets and introducing errors into a few of them as specified by the input of the algorithm. The proposed methods will allow the detection of attacks on a subset level of state estimation variables.
- **Machine learning-based detection methods.** State-of-the-art machine learning models will be proposed for the detection of attacks on state estimation, especially with regard to subset level detection. Due to the naturally occuring temporal dependencies in the data, a special focus will be on Recurrent Neural Networks (RNNs) using Long Short-Term Memory (LSTM) units. The performance of the models in detecting attacks will be evaluated by varying the number of subsets, the strength of attacks and different preprocessing methods.
- **Implementation is open source.** All software that is developed in the context of this thesis will be made public under an open-source license model.

## 1.2 Description of the Remaining Chapters

Chapter 1 outlines the motivation and objectives of this thesis, gives a general introduction and an overview of the remaining chapters.

All background, except of machine learning related background, is defined in Chapter 2. This chapter starts off by outlining cyber-physical systems with a special focus on privacy and security and highlights a specific type of cyber-physical systems, the so-called intelligent energy networks. Basic terminology used in the power grid and the concept of both state estimation and False Data Injection Attacks (FDIAs) are defined. The purpose of this chapter is to form a solid basis for the remaining parts of this thesis.

An extensive literature review, including all major works to the best of the author's knowledge up to the time of writing, is found in Chapter 3. Research works investigating both the construction of FDIAs, the detection of FDIAs and the protection against FDIAs are considered. Machine learning-based detection methods are covered in more detail.

Chapter 4 starts by outlining the concept of machine learning, introducing basic terminology and different machine learning schemes. The supervised learning models Support Vector Machines (SVMs), Neural Networks (NNs) and Recurrent Neural Networks (RNNs) are investigated, due to the use of these models in the practical part of this thesis. Different types of preprocessing methods, as well as performance metrics with a special focus on imbalanced classes, are highlighted.

In Chapter 5, an overview of the proposed test framework and its individual components, is given. The practical generation of data sets based on real-world load data is described. Furthermore, the construction of attacks targeting all state estimation variables or only subsets of state estimation variables are explained. Different types of attack strength are considered and analyzed.

Chapter 6 represents the practical part of this thesis, containing all conducted experiments. First, the experimental setup and the proposed machine learning models are described. FDIAs with different types of attack strength and energy are evaluated and conclusions are drawn. To increase performance, multiple preprocessing methods are cross-validated on the proposed models and the best methods determined. Each proposed model and its best performing preprocessing model is then evaluated on detecting attacks targeting different numbers of subsets.

Finally, Chapter 7 contains a summary of this thesis. The main achievements are discussed and connections to the main objectives, that were defined in Chapter 1, are drawn. This thesis concludes by discussing possible future work and new problems that can be defined based on the research findings in this thesis.

# Chapter 2

# Background

This chapter gives an overview of the background required for the understanding of the remaining part of this thesis. First, cyber-physical systems are investigated with a focus on security and the construction of secure architectures. A special type of cyber-physical systems, the intelligent energy networks, known as smart grids, are highlighted. Subsequently, basic terminology used in the power grid is defined and the concept of state estimation introduced. Static state estimation for both AC and DC power flow models is explained. The need for a BDD is emphasized and a specifically fatal type of attacks on state estimation, the so-called False Data Injection Attacks (FDIAs) are mentioned. Concrete examples are given throughout the entire chapter.

## 2.1 Cyber-Physical Systems

Technology is constantly finding its way into more and more areas of everyday life. The World-Wide Web (WWW) made it possible to connect both devices as well as human beings from all over the world, improving collaboration, sharing ideas and working together more efficiently and faster. Technological progress such as the wireless communication revolution, the possibility to manufacture highly powerful sensors and chips at low cost and the improvement in energy storage technology, paved the path for a new generation of systems, the so-called *cyber-physical systems*.

Cyber-physical systems bridge the gap between the physical world and the cyber-world, allowing physical systems to be controlled, monitored and integrated by a computing environment. According to Lee in [2], cyber-physical systems have the

potential to revolutionize IT development over the next few decades, improving and enabling applications in medical systems, transport and autonomous systems such as autonomous driving, energy, safety, public infrastructures and more. Jazdi [7] investigated the influence of cyber-physical systems in the context of Industry 4.0, where automated systems can help to improve both efficiency as well as security of certain manufacturing processes. Although bridging the gap between physical and digital systems enables new functionality, it must not be forgotten to also consider both security and privacy when drafting and implementing such new systems. As stated by Cardenas *et al.* in [8], cyber-physical systems require additional, fundamentally different security considerations compared to traditional IT systems. One example for a major challenge is the process of patching software and the frequent installation of updates in control systems. Due to the fact that taking a system offline could require several months of preparation, upgrades have to be scheduled carefully. Another challenge is dealing with the vast amount of legacy systems that are usually still present in large industrial control installations. On the positive side, however, network dynamics in control systems are usually simpler, thus making it easier to install network intrusion detection systems due to a more static topology.

The next section discusses a special type of cyber-physical systems, namely the smart grid. As stated by Baheti *et al.* [9], smart grid and renewable energy research is in high public interest. One of the major goals in this field is the reduction of energy costs by improving energy efficiency. The authors also forecasted that electricity demand is expected to increase by more than 75% until 2030.

### 2.1.1 Smart Grid

Energy is a vital part of our modern society and its absence can lead to severe socio-economic damage. As described in [10], economic growth and energy consumption rise proportionally. Moreover, power plants based on renewable energy such as solar power, wind, water and biomass, are deployed more frequently, thus causing a dynamic behaviour of power flow in the power grid [11]. In addition to dynamic generation, electric vehicles and ever more sophisticated charging processes are creating dynamic demands. To ensure stability in the power grid, the balance between supply and demand has to be maintained at all times. In order to cope with these upcoming challenges, a new type of grid, namely the *smart grid*, has formed. Whereas the purpose of the traditional power grid was to transfer energy from a few central generators to a varying number of customers, the smart grid enhances this functionality by enabling a two-way

flow of both electricity and information [3]. This enhancement makes it possible to construct more dynamic, distributed and automated energy delivery networks. One key feature of smart grids is that they will be self-healing [12]. In the case of a transmission line interruption, power flow can be automatically redirected and adjusted, therefore preventing a power outage.
According to Fang *et al.* [3], the smart grid can be divided into the following three fundamental systems, which give a high-level overview of the smart grid from a technical point of view:

1. **Smart infrastructure system.** The smart infrastructure system is the base layer of the smart grid, providing an infrastructure for energy, information and communication flow. The first component, the *smart energy subsystem*, consists of advanced electricity generation, delivery and consumption, whereas the task of the *smart information subsystem* is to provide advanced metering, monitoring and management information. One major component of the smart information subsystem is the AMI [13], which gathers all necessary information and data from the smart grid by use of smart meters and sensors. The latter, the *smart communication subsystem*, is responsible for a reliable communication connectivity between the devices in the smart grid.
2. **Smart management system.** The smart management system provides core management services and functionalities. By implementing new complex and sophisticated management applications and algorithms, this system can allow the smart grid to become even more intelligent.
3. **Smart protection system.** The smart protection system addresses grid infrastructure failures due to various reasons, such as user errors, natural disasters or cyber-physical attacks. Both privacy as well as security aspects have to be taken into consideration, when developing applications for the smart protection system. Moreover, IDSs and Intrusion Prevention Systems (IPSs) are part of this fundamental building block. Because the focus of this thesis is on security and protection, the smart protection system will be investigated in more detail below.

To avoid power outages which may lead to severe socioeconomic damage, protection is of utmost importance. An example is given by Moslehi *et al.* in [14], who stated that the annual cost of outages in the United States in 2002 was estimated to be around 79 billion dollars, whereas the total retail revenue was 249 billion dollars. Another example of the past is the cascading blackouts incident on the East Coast in 2003,

which caused around 50 million people to be without power for up to several days [15]. Due to the complex nature of a smart grid, a smart protection system needs to account for numerous different attack vectors in order to provide protection for all parts of the smart grid.

Fang *et al.* [3] differentiated between the following two parts in smart protection systems: i) *system reliability and failure protection* and ii) *security and privacy.*

*System reliability* refers to the ability of a system to operate under specified conditions for a predetermined amount of time. One concept that may have an impact on reliability is distributed generation, which has been investigated in several works. In [16], Chen *et al.* introduced the concept of *microgrids* to minimize the likelihood of cascading failures and effectively utilize distributed generation sources. The reliability of the measurement system, which is used to monitor the operating conditions of the smart grid, is important as well.

When it comes to *failure protection*, it is both necessary to protect (i.e., prevent) the system from becoming faulty, as well as being able to identify and diagnose a failure and recover once a failure occurred. At this point the self-healing aspect of the smart grid comes into view again. In [17], Chertkov *et al.* developed a concept based on worst configuration heuristics to predict weak points of the power grid that need to be protected. In [18], Tate *et al.* made use of Phasor Measurement Units (PMUs), which are becoming increasingly widespread, in order to detect line outages and estimate the pre-outage flow on the outaged line. By dividing the power grid into smaller islands, the previously mentioned microgrids, Rahman *et al.* [19] proposed an architecture that allows the smart grid to become more resilient and enables the normal operation within a microgrid even during an outage.

Another major research area in the smart grid is *security and privacy.* On one hand the advanced infrastructure enables the use of better methods to defend against attacks, but on the other hand makes the system more vulnerable. Security and privacy have to be considered both in information metering and measurement as well as information transmission. One popular target for attackers are smart meters. If a smart meter is compromised, it may be easy for an attacker to manipulate meter readings. On a small scale, this could be exploited to communicate a lower energy consumption to the system operator [20]. In [21], Anderson *et al.* gave an example with much more impact, where they assumed, that an arbitrary city installs millions of smart meters with bad security design, which are controlled by a single head-end. An attacker that is able to compromise the head-end may send a command to the smart meters, instructing

Table 2.1 Estimated maximum latency per communication type [22].

| Maximum Latency | Communication Type |
|---|---|
| <4ms | Protective relaying |
| Sub-seconds | Wide area situational awareness monitoring |
| Seconds | Substation and feeder supervisory control and data acquisition (SCADA) |
| Minutes | Monitoring noncritical equipment and marketing pricing info |
| Hours | Meter reading and longer-term pricing info |
| Days/Weeks/Months | Collecting long-term usage data |

them to disconnect the power supply and change the crypto keys to some value that is only known to the attackers. Both examples illustrate the impact of attacks on cyber-physical systems such as the smart grid and the necessity for proper security and privacy implementations. Baumeister [12] stated that the most important security objective of the smart grid is availability, followed by integrity and confidentiality. The security objective availability is broken down into more detail in Table 2.1, where the maximum allowed latency per communication type is displayed, as specified by the National Institute of Standards and Technology (NIST) [22].

Baumeister [12] split smart grid security into five categories, which he also used as classification for research work in this area. The five categories are outlined below:

1. **PCS security.** Process Control Systems (PCSs) are used to monitor and control physical aspects of the smart grid. Due to the fact that PCSs have not been connected to the internet in traditional power networks, security was not of major concern. This needs to be changed when it comes to a smart grid. The most important objectives of a PCS are availability and integrity, because a working and available PCS is the most important requirement for a functioning electric power system. Due to the added overhead when implementing confidentiality, this security objective is of least importance. Although different kind of PCSs exist, the most commonly used is the Supervisory Control and Data Acquisition (SCADA) system.
2. **Smart meter security.** Smart meters are electronic versions of the traditional power meters that communicate energy usage back to the energy suppliers at regular intervals. Due to the fact that they are installed at a customer's site, it is harder to enforce security because physical access to the smart meter can be gained easily. Tampering smart meters for monitory gain is a prominent attack vector. McDaniel *et al.* stated in [20], that energy amounting to approximately six billion dollars was stolen from the U.S. electric power system. Confidentiality

and integrity are the most important security objectives in this area. Equally important is the privacy aspect. Methods to estimate personal information based on power usage information have already been developed [23].

3. **Power system state estimation security.** State estimation is used to model the current state of the power system in order to make decisions and take appropriate actions. Correct state estimation is needed to take correct mitigation actions in case of power outages. Similar to PCS security, availability and integrity are the most important security objectives, whereas confidentiality is neglected due to the additional communication overhead. In practice, state estimation is implemented as a module in PCSs, which are usually equipped with an additional BDD that filters out incorrect and faulty measurements. It has been shown however, that there exist specific attacks that can bypass this BDD [24]. In later parts of this thesis, different attacks and corresponding detection and protection schemes are investigated.

4. **Smart grid communication protocol security.** The purpose of this area is to develop and implement security requirements for the communication protocols used in smart grids. As mentioned previously in this section, many different devices need to interact with each other to transmit metering, monitoring and management information. The security objectives depend on the type of devices that are communicating. As stated by Lu *et al.* [25], it is important to ensure network availability, data integrity and information privacy. They proposed an authentication protocol design and intrusion detection for a more secure and private information transmission.

5. **Smart grid simulation for security analysis.** The last section includes smart grid simulation. The physical power grid cannot be modified arbitrarily, thus simulation methods are used to develop and test different algorithms and methods and analyze security aspects. One major challenge is the creation of a simulation that is as close to the real-world behaviour of smart grids as possible. Another challenge is that specific components may act similar in the simulation, but have unpredictable behaviour in real-world smart grids, which are hard to model in the simulation.

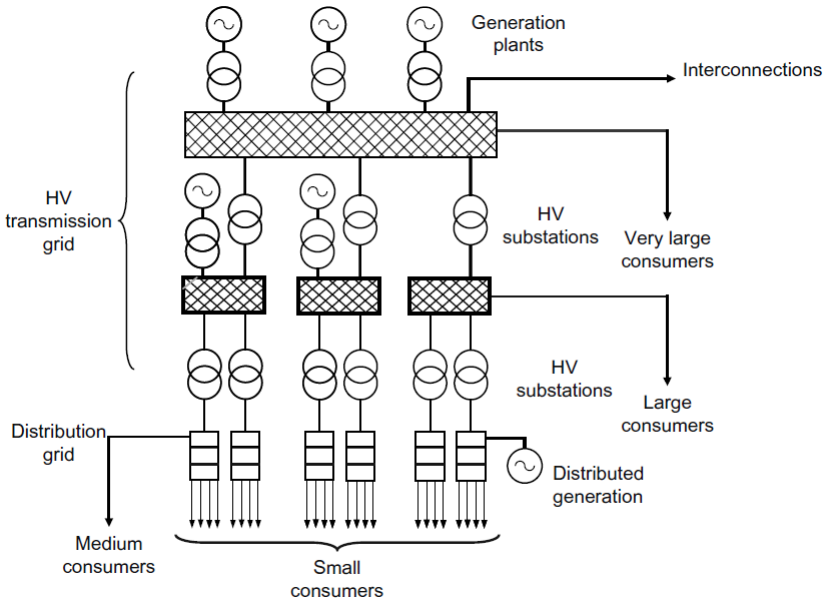



Figure 2.1 Structure and configuration of an electric power system [5].

## 2.2 Power Grid

Before introducing the concept of state estimation, basic terminology and methodology used in power grids must be defined. This section provides an overview and concludes by showing how a power grid can be transformed into a graph-based representation, which is oftentimes required in order to apply graph-theoretical concepts.

The basic structure and configuration of an electric power system, as usually implemented by utility companies all over the world, is illustrated in Figure 2.1 [5].

As stated in [5], the transmission network is used to transfer power from the power generators to the distribution network. To minimize losses during power transmission due to lossy transmission lines, electricity that is generated by the power generators in the range of several kilovolts, typically 6 to 20 kV, is immediately transformed to voltages in the range of several hundred kilovolts. Large end-consumers are connected directly to the transmission network, whereas small customers are connected to the distribution network. In both cases, voltage transformers are used to convert the high voltages in the transmission network to smaller voltages. Voltages in the distribution network, usually around 15 or 20 kV, are lower in order to be less dangerous due to the use in urban areas. In order to maintain stability in the power grid, it has to be ensured that the frequency is kept at a fixed value (e.g., 50 Hertz in the European power grid) and only deviates within a specified range. This frequency must be maintained at all

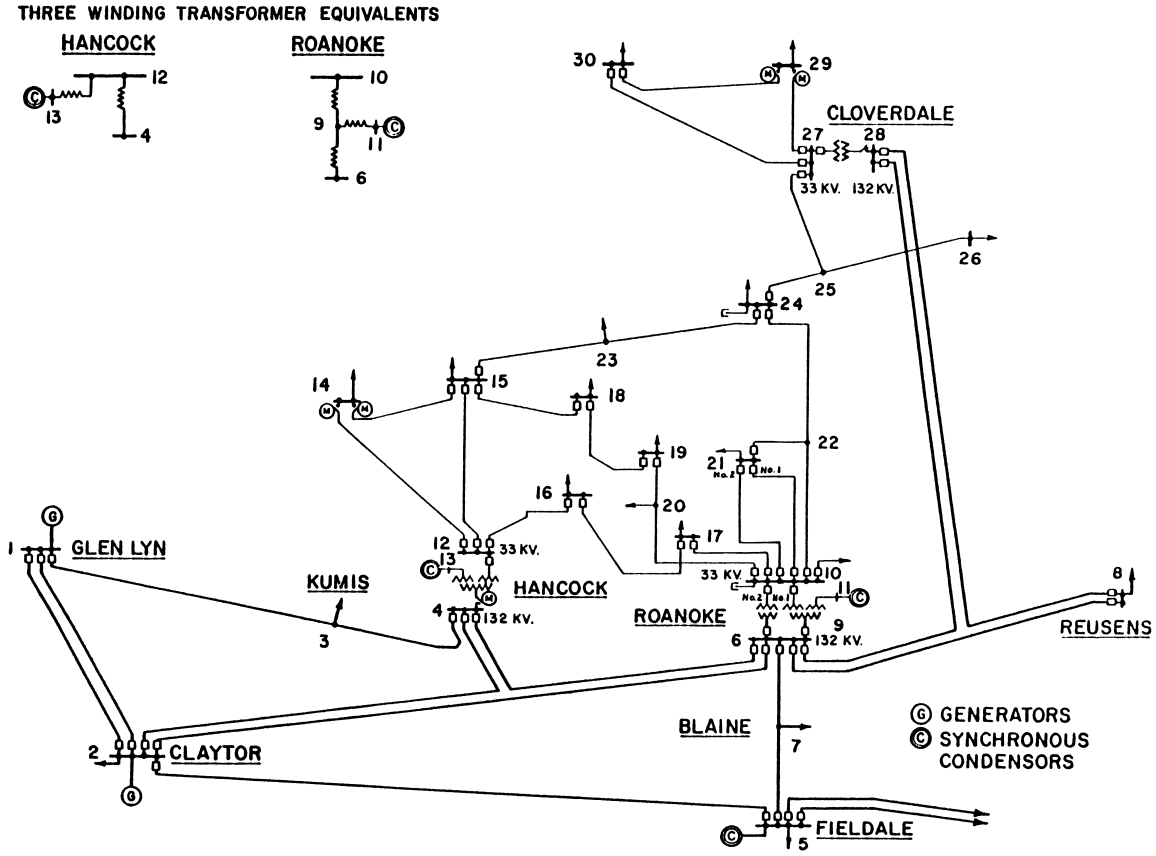



Figure 2.2 IEEE 30 bus test case [27].

times in the entire power grid, ranging from generators at power generation plants up to computers and coffee machines of end-consumers.

As mentioned previously in Section 2.1.1, new algorithms and concepts are often tested using simulations of the power grid. Due to the fact that exact models of the power grid are often hard to obtain for researchers, the University of Washington created a Power Systems Test Case Archive [26], containing different power flow test cases. These test cases are broadly used in various research works as basis for simulations. It is further beneficial to use similar power flow test cases, as the performance of different algorithms can be compared against each other. All power flow test cases are available in different standardized data formats. Figure 2.2 [27] displays the broadly used IEEE 30 bus test case, which represents a portion of the American Electric Power System as of December, 1961. This test case will also be used later in this thesis.

By analyzing Figure 2.2, it can be seen that the electric network consists of a set of *branches* and *buses*, whereby each branch is connected to two *terminal buses*, which may be again connected to one or more different branches in the network. If one of

the terminal buses is connected to ground, the corresponding branch is referred to as *shunt branch*. Regarding the previously mentioned power flow test cases, all electrical parameters of the branches and buses are known and documented. In [28], all available parameters of the power flow test cases are found, according to the IEEE Common Data Format.

Another important concept in electric systems is the *power flow problem*. As described in [5], the power flow problem is defined as finding the steady-state operating point of the electric power system. It is the most basic tool for security analysis during the daily grid operation for power grid providers. By continuously calculating the steady-state operating point using measurements in the power grid, the power grid provider can easily identify unacceptable voltage deviations and component or branch overloadings. Knowledge of load or generation at all buses, except the reference bus, is required. As mentioned by Wood *et al.* [29], for the ease of computational complexity, power flow studies rely on a single-phase per-unit representation of the power grid instead of the usual three-phase Alternating Current (AC) network. This is only possible when the system is operating in a balanced mode, meaning that the current of all phases is shifted by $\pm 120°$ and the voltage is the same.

It is often beneficial to convert the power grid model to a graph-based representation. This is done by replacing the branches with either undirected or directed edges and the buses with nodes. Considering the power grid as a graph enables the use of graph-theoretical approaches, which can help in more efficiently finding solutions to problems such as topological observability analysis [30] or the identification of weak spots [17], as mentioned earlier in Section 2.1.1. Additional benefits of displaying the power grid as a graph will be discussed in Chapter 3.

## 2.3 State Estimation

State estimation in the power grid dates back until 1970 when it was first introduced by Schweppe in [31]. Since then it has become a core functionality for supervisory control and planning in electrical power grids [32], usually being implemented in SCADA systems. State estimation has a long history in transmission networks, while further applications and developments are still under research. As mentioned by Huang *et al.* [32], research during the last four decades was mainly focused on *static state estimation*. This was due to the fact that monitoring systems were only able to take non-synchronized measurements every few seconds (e.g., 2-4 seconds) and state estimation was run only

every few minutes. Because of that limitation, this type of state estimation is considered to be static and not dynamic.

As technology progressed, more sophisticated measurement technologies such as the Phasor Measurement Unit (PMU) were developed and their applicability for state estimation investigated. PMUs are synchronized to a Global Positioning System (GPS) clock and allow a more timely view of the dynamics of a power grid. Zhou *et al.* in [33] mentioned, that PMUs can be used to measure the system state instead of estimating it. One difficulty of this approach is the large number of PMUs that have to be deployed, thus several authors such as Zhou *et al.* [33] and Phadke *et al.* [34] investigated the advantages of utilizing PMUs to improve the results of state estimation.

Although this thesis will focus on static state estimation, it must be mentioned that other concepts of state estimation, which might play an important role in a smart grid that is becoming ever increasingly dynamic [32], were proposed too. An example for a dynamic state estimation architecture is *forecasting-aided state estimation*, which not only relies on all measurements taken as a single snapshot at a specific time, but also provides a recursive update of the state estimate to track changes that occur during normal system operation. *Multiarea state estimation* tries to reduce computational complexity by dividing one large area into several smaller ones, providing several local solutions for several small areas which are then combined into one global state estimate. Another active research area is *distribution system state estimation*, where state estimation is applied to the distribution network of the power grid which poses an inherently different nature than the transmission network.

In previous parts, different types of state estimation architectures were discussed. The next part explains the concept of static state estimation in the power grid and gives a formal introduction.

A smart grid consists of an AMI, multiple power plants and substations as well as communication networks to communicate sensor readings, control information and distributed power generation. All data from the communication network is then processed at SCADA/EMS to continuously monitor the operating conditions of the power grid. Figure 2.3 illustrates the relationship between the single elements of a SCADA/EMS.

Data is collected by Remote Terminal Units (RTUs) and, more recently, Phasor Data Concentrators (PDCs). The topology processor keeps track of the current topology of the power grid. To be able to perform state estimation, the network has to be observable, which is determined by the observability analysis component. The bad-data

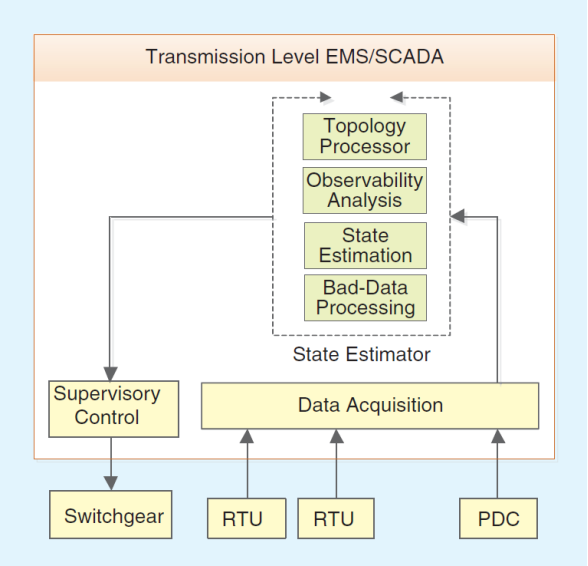



Figure 2.3 Elements of a SCADA/EMS [32].

processor removes faulty or incorrect measurements prior to state estimation. State estimation is then used to convert meter measurements, that are often redundant by nature, into a single system state using power flow models.

The system state of the power grid can be entirely described by the voltage magnitudes and phase angles at every bus and the topology and impedance of the entire system [32]. The following list gives an overview of the most common types of measurements in a power grid, as illustrated in [5]:

1. Real and reactive power flows measured at buses.
2. Net real and reactive power injections measured at buses.
3. Voltage magnitudes measured at buses.
4. Current magnitudes (i.e., ampere flows) measured at buses.

The formal solution to a state estimation problem differs when considering either an AC or a Direct Current (DC) network. Next, static state estimation in AC networks is described followed by static state estimation in DC networks. A short example using a three bus system for DC state estimation is given.

### 2.3.1 AC State Estimation

Considering an $N$-bus power grid, the state vector consists of $n = (2N - 1)$ state variables, resulting in a vector of form $x = [\theta_2, \ldots, \theta_N, |V_1|, \ldots, |V_N|]^T$, where $\theta_i$ are the phase angles and $V_i$ the voltage magnitudes at the buses. $\theta_1$ is the phase angle of the reference bus which is known and usually set to zero radians. To be able to estimate the state $x$, a set of measurements $z = [z_1, \ldots, z_m]^T$ has to be obtained. If $m > n$, that is the number of measurements is larger than the number of state variables, the system is overdetermined and a solution can be found. The relationship between the state vector and the vector of measurements is given by the following system of nonlinear equations [31]:

$$z = h(x) + e \tag{2.1}$$

where $h_i(x)$ is the nonlinear function relating measurement $i$ to the true state vector $x$ and $e = [e_1, \ldots, e_m]^T$ is the vector of measurement errors. The measurement errors are usually considered to be independent, normally distributed and with an expected value of zero. The following statement summarizes these characteristics:

$$Cov(e) = E[e \cdot E^T] = R = diag\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2\} \tag{2.2}$$

In the traditional approach by Schweppe in [31], a Weighted Least Squares (WLS) estimator is used to estimate the state vector $\hat{x}$ from the measurement equation in 2.1. Because the measurement errors are considered to be normally distributed, the WLS estimator is equivalent to a maximum likelihood estimator. The remainder of this section outlines the approach of solving the AC state estimation problem. The proof and mathematical details are omitted for brevity, but can be found in [31].

The WLS estimation problem can be formulated as follows:

$$J(x) = \underset{x}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{[z_i - h_i(x)]^2}{\sigma_i^2} \tag{2.3}$$

which can be rewritten to a more compact form by placing all $h_i$ functions and $z_i$ measurements in a vector:

$$J(x) = \underset{x}{\operatorname{argmin}}[z - h(x)]^T W[z - h(x)] \tag{2.4}$$

where $W$ is a diagonal matrix whose elements are reciprocals of the meter errors $\sigma_i^2$. The elements of $W$ can be thought of as weights assigned to individual meter measurements. To find a solution to the optimization problem in Equation 2.4, the following optimality conditions have to be satisfied:

$$\frac{\partial J(x)}{\partial x} = 0 \implies H^T(x)W[z - h(x)] = 0 \tag{2.5}$$

where

$$H(x) = \frac{\partial h(x)}{\partial x} \tag{2.6}$$

is the $m \times n$ Jacobian matrix of the first-order partial derivatives of $h(x)$ with regard to $x$.

Because Equation 2.4 is nonlinear due to the AC state estimation, the solution cannot be found in closed form but has to be obtained in an iterative approach. The most common way is to apply the Newton-Raphson (NR) iterative process, as described in [31] and [35], to improve the estimate $x$ at every iteration $k$ and converge to the solution. The equation that needs to be solved at every iteration $k$ is given by:

$$G(x^k)\Delta x^k = H^T(x^k)W[z - h(x^k)] \tag{2.7}$$

where $x^k$ describes the value of $x$ at each iteration $k$ and

$$G(x) = H^T(x)WH(x) \tag{2.8}$$

is known as gain matrix. Before the process is repeated, the estimated state vector is updated:

$$x^{k+1} = x^k + \Delta x^k \tag{2.9}$$

Equation 2.7 is also known as normal equation.

After each iteration, the norm of the residual $||z - h(x^k)||^2$ is evaluated. Once the norm of the residual falls below a predefined value, that is for some $\delta > 0, ||z - h(x(j))||^2 \leq \delta$, the process is terminated and the estimator is considered to be converged.

If $\hat{x}$ defines the converged state estimation vector, the vector of measurement residuals after WLS state estimation is given by the following equation:

$$r = z - \hat{z} = z - h(\hat{x}) \tag{2.10}$$

There exist several methods to reduce the computational complexity of solving the normal equation. One of them is dividing the problem into the active and reactive subproblem, which leads to the decoupled estimators, as described by Garcia *et al.* in [36] and Wang *et al.* in [37]. Another option is to utilize the sparseness of the matrices $G$ and $H$ and implement better suited iterative solutions, such as the Krylov subspace methods [32].

At the beginning of the iterative process, the state vector $x$ is usually initialized using a flat voltage profile, i.e., $V_i = 1$ p.u. (per unit) and $\theta_i = 0$.

### 2.3.2 DC State Estimation

According to Monticelli in [38], state variables can be defined as any set of variables that fulfill the following two characteristics: i) state variables describe the system entirely, meaning that when all state variables are known, all other remaining variables can be derived using the network model equations; ii) if any of the state variables is removed from the set, the property in i) does not hold. Although the author stated that any measurement variable can be used as a state variable, voltage magnitudes and reactive power flows usually are of little concern in DC state estimation. Thus, only phase angles at the buses are considered as state estimation variables, similar to Liu *et al.* in [24]. The following list summarizes the criteria for DC state estimation [39]:

- The linear DC approximation model for the measurement equations is used.
- All system branches have an impedance of $1.0j$ p.u. and all bus voltage magnitudes are close to 1.0 p.u..
- The state vector consists of only phase angles, that is $x = [\theta_2, \ldots, \theta_N]$, where $\theta_1$ is the phase angle of the reference bus and is set to zero radians as usual.

- Voltage angle differences between branches are small, such that $\sin(\delta\theta) \approx \delta\theta$.
- Only real, as opposed to real and reactive, measurements are considered.
- All shunt elements and branch resistances are neglected.

Due to the aforementioned criteria, the nonlinear function in Equation 2.1 can be rewritten into the following linearized measurement model:

$$z_A = H_{AA}\theta + e_A \tag{2.11}$$

where $z_A$ includes real power flows and power injections measured at buses, $H_{AA}$ is a function relating the measurements to the state estimation variables consisting of branch reactances only, and $e_A$ is the vector of measurement errors, as already described in AC state estimation.

Because of these assumptions, the functions $h(x)$ in Equation 2.4 are now linear, too. Thus, a solution to the optimization problem can be found in closed form as opposed to the iterative approach in AC state estimation. By satisfying the conditions stated in Equation 2.5, the solution is given by:

$$\hat{x} = (H^T W H)^{-1} H^T W z \tag{2.12}$$

Mathematical proof and numerical details on how to solve the optimization problem are omitted for brevity, but can be found in [29]. It has to be noted, that Equation 2.12 only holds true, when the system is overdetermined, that is $m > n$. When the system is completely determined, the estimation problem reduces to the simplified form

$$\hat{x} = H^{-1} z \tag{2.13}$$

When the system is underdetermined, that is $m < n$, the closed-form solution leads to a different objective. In state estimation, underdetermined systems are usually not solved, but *pseudo-measurements* are added to the measurements in order to make the system completely determined or overdetermined.

In addition to the estimate of the vector, an estimate $\hat{z}$ of the measurement vector can be obtained by:

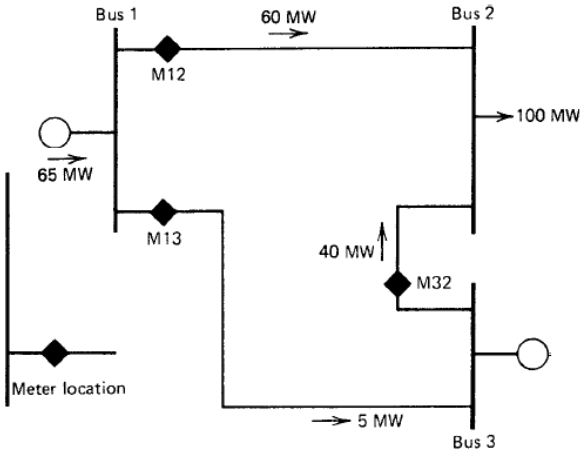



Figure 2.4 Topology of the 3 bus system [29].

$$\hat{z} = H\hat{x} \tag{2.14}$$

The vector of residuals is defined as:

$$r = z - \hat{z} = z - H\hat{x} \tag{2.15}$$

Next, a simple state estimation problem with redundant measurements using a 3-bus system, similar to the one described in [29], is given.

Consider the 3-bus system illustrated in Figure 2.4, where flow meters are placed on each branch. The number of measurements is $m = 3$. The state variables are represented by the phase angles at the buses except the reference bus, i.e., $x = [\theta_2, \theta_3]^T$, resulting in $n = 2$. Because of $m > n$, the DC state estimation problem is overdetermined and can be solved in closed form. Let's assume that bus 3 is the reference bus with a known phase angle set to zero radians, i.e., $\theta_3 = 0$. The branch reactances, given a 100 MVA base, are per unit and defined as follows:

$$X_{12} = 0.4, X_{13} = 0.8, X_{23} = 0.5$$

In order to derive the measurement matrix $H$, the measurements have to be written as a function of the state variables $\theta_1$ and $\theta_2$. Because $\theta_3$ is assumed to be zero, this results in the following set of equations:

$$
\begin{aligned}
M_{12} = f_{12} &= \frac{1}{0.4}(\theta_1 - \theta_2) = 2.5\theta_1 - 2.5\theta_2 \\
M_{13} = f_{13} &= \frac{1}{0.8}(\theta_1 - \theta_3) = 1.25\theta_1 - 1.25\theta_3 = 1.25\theta_1 \\
M_{32} = f_{32} &= \frac{1}{0.5}(\theta_3 - \theta_2) = 8\theta_3 - 8\theta_2 = -2\theta_2
\end{aligned} \tag{2.16}
$$

$H$ can now be constructed using the previously determined coefficients:

$$
H = \begin{bmatrix} 2.5 & -2.5 \\ 1.25 & 0 \\ 0 & -2 \end{bmatrix}
$$

The matrix of meter errors $R$ and the corresponding matrix $W$ is:

$$
R = \begin{bmatrix} \sigma^2_{M12} & & \\ & \sigma^2_{M13} & \\ & & \sigma^2_{M32} \end{bmatrix} = \begin{bmatrix} 0.0001 & & \\ & 0.0001 & \\ & & 0.0001 \end{bmatrix}
$$

$$
W = R^{-1} = \begin{bmatrix} 10000 & & \\ & 10000 & \\ & & 10000 \end{bmatrix}
$$

Since everything is written in per unit, the measurements have to be written in per unit as well. This is achieved by dividing the measurement value by the base load, which is 100 MVA in this case. Populating Equation 2.12 with the given values results in the following least-squares estimate of $\theta_1$ and $\theta_2$:

$$\hat{x} = \left[ \begin{bmatrix} 2.5 & 1.25 & 0 \\ -2.5 & 0 & -2 \end{bmatrix} \begin{bmatrix} 10000 & & \\ & 10000 & \\ & & 10000 \end{bmatrix}^{-1} \begin{bmatrix} 2.5 & -2.5 \\ 1.25 & 0 \\ 0 & -2 \end{bmatrix} \right]^{-1}$$
$$\times \begin{bmatrix} 2.5 & 1.25 & 0 \\ -2.5 & 0 & -2 \end{bmatrix} \begin{bmatrix} 10000 & & \\ & 10000 & \\ & & 10000 \end{bmatrix}^{-1} \begin{bmatrix} 0.62 \\ 0.06 \\ 0.37 \end{bmatrix}$$
$$= \begin{bmatrix} 0.0571 \\ -0.1886 \end{bmatrix}$$

Now that the state estimation variables are obtained, the power system is completely describable and power flow in each transmission line as well as net generation or load at each bus can be calculated. Calculating power flows using the state variables results in the following:

$$M_{12} = 61.4MW, M_{13} = 7.14MW, M_{32} = 37.7MW$$

The deviation (e.g., $M_{12}$ is 0.62 instead of 0.6) is due to the slight errors in the measurements that were assumed during state estimation.

As could be seen, static state estimation is used to convert redundant and sometimes defective meter measurements into a single, consistent system state. The next section introduces standard methods to deal with bad data such as incorrect and faulty measurements.

### 2.3.3 Bad Data Processing

Meters in a power grid may be analogue or digital and may contain measurement errors according to their accuracy. Moreover, meter readings may be subject to noise when being transmitted via the telecommunication channels to the central management location. Sometimes, even meter connections may be changed without report of those changes to the supervisory system, thus yielding unexpected results. All those measurement errors occur naturally and have to be accounted for. As stated in [5], accuracy and reliability of the state estimator depend on the quality of the measurements. In order to obtain correct results, bad measurements have to be filtered out prior to

state estimation. This is usually done by a so-called *bad-data processor*. In the case of WLS state estimation, bad data is processed after state estimation by analyzing the measurement residuals. In general, it is distinguished between bad data detection and bad data identification, whereby the former detects if bad data is present and the latter classifies which measurements are faulty. Moreover, it has to be noted, that bad data detection and identification is only possible when redundant measurements exist. If $m = n$, there are no redundant measurements and the removal of a single measurement would render the system unobservable.

Different bad data processing methods exist. Two of the more prominent methods for detecting and identifying bad data after the WLS state estimation, the $\chi^2$ test and the Largest Normalized Residual ($r^N_{max}$) test as described in [40], are discussed in more detail below. The former is strictly used for bad data detection, whereas the latter can also be used for bad data identification. Both methods are applicable to AC as well as DC state estimation. Additionally, there exist methods that deal with bad data during the state estimation process, as described by Merrill *et al.* in [41].

### Bad Data Detection

The $\chi^2$ test is based on the assumption that an index $J(\hat{x})$ follows a Chi-square distribution with $m - n$ degrees of freedom, when meter errors are assumed to be normally distributed and no bad data is present. The computed value of $J(\hat{x})$ is then compared to a constant calculated from the Chi-square distribution. If $J(\hat{x})$ exceeds the calculated value, it is assumed that bad data is present. Equations 2.10 and 2.15 show the calculation of the measurement residuals. The $\chi^2$ test can then be summarized by the following procedure:

1. Solve the state estimation problem and compute the objective function $J(\hat{x})$:

$$J(\hat{x}) = \sum_{i=1}^{m} \frac{[z_i - h_i(\hat{x})]^2}{\sigma_i^2}$$

   where $\hat{x}$ is the estimated state vector of size $n \times 1$, $m$ the total number of measurements, $h_i(\hat{x})$ the estimated measurement $i$, $z_i$ the true measured value of measurement $i$ and $\sigma_i^2$ the variance of the measurement $i$.

2. Determine the constant $\chi^2_{(m-n),p}$ by looking up the value from the $\chi^2$ distribution table, given the probability $p$ and the degrees of freedom $m - n$.

3. Test if $J(\hat{x}) \geq \chi^2_{(m-n),p}$. If yes, bad data is assumed to be present by a probability of $p$. If not, it is assumed that the measurements are free of bad data.

### Bad Data Identification

dThe Largest Normalized Residual ($r^N_{max}$) test gives the advantage that bad data cannot only be detected, but also identified. By normalizing the residuals it can be shown, that the largest residual corresponds to the bad data if a single bad measurement is present in the measurement set. This also applies to multiple bad measurements, if none of them interact with each other. The procedure of detecting and identifying single or multiple non-interacting bad data is described as follows:

1. Solve the state estimation problem and obtain the measurements residual vector $r_i \in \mathbb{R}^m$ according to Equation 2.10 in the case of AC state estimation or Equation 2.15 in the case of DC state estimation.
2. Compute the normalized residuals:

$$r^N_i = \frac{|r_i|}{\sqrt{\Omega_{ii}}}, \quad i = (1, \ldots, m)$$

   where $\Omega_{ii}$ is the $ii$th element of the residual covariance matrix $\Omega$.
3. Find the largest element $r^N_k$ among all $r^N_i$, $\quad i = (1, \ldots, m)$.
4. If $r^N_k$ is larger than a predefined threshold $c$, the $k$th measurement is assumed to be bad data. Remove the measurement from the measurement set and repeat with step 1. Else, stop, no more bad data will be suspected. According to [10], $c$ is usually chosen to be 3.

### Classification of Bad Data

Bad data can exist as either *single bad data* or *multiple bad data*. Single bad data is present, if and only if exactly one measurement in the set of measurements is faulty. Otherwise, multiple bad data is present, which can furthermore be divided into *non-interacting* and *interacting* as well as *conforming* and *non-conforming* bad data. To be able to distinguish between the aforementioned types, the *residual sensitivity matrix* $S$ is needed. Following, only the calculation of $S$ based on the linearized DC measurement

model is defined, because it is sufficient for the remaining part of this thesis. The derivation is omitted for brevity but can be found in [10].

$$\begin{aligned} S = (I - K) &= (I - H(G)^{-1}H^T W) \\ &= (I - H(H^T W H)^{-1}H^T W) \end{aligned} \tag{2.17}$$

The following listing summarizes the effectiveness of the $r^N_{max}$ test with regard to the different types of bad data:

- **Single bad data.** When single bad data is present, the test will correspond to the bad measurement.
- **Non-interacting multiple bad data.** It is said that measurements $i$ and $k$ are non-interacting when $S_{ik} \approx 0$. In this case, the test is able to identify both faulty measurements sequentially, i.e., one bad data per iteration.
- **Interacting, non-conforming multiple bad data.** If $S_{ik}$ is very large, it is said that both measurements are interacting. The test is still able to detect bad data correctly, if the errors in measurements $i$ and $k$ are not consistent with each other.
- **Interacting, conforming multiple bad data.** In this special case the test may fail to identify either of the both.

It has to be noted, that the $r^N_{max}$ test only gives good and reliable results when the measurement residuals are not strongly correlated. One approach to tackle the issue of strongly correlated measurement residuals is to use the Hypothesis Testing Identification (HTI) method, as elaborated by Mili *et al.* in [42] and [43], who tried to distinguish between good and bad measurements by estimating the measurement errors directly, instead of relying on the derived measurement residuals.

## 2.4 False Data Injection Attacks

In Section 2.3, the concept of state estimation and its importance for modern utility companies was explained. Moreover, traditional bad data processing was investigated, outlining two prominent methods for both bad data detection and bad data identification.

This section introduces a specific type of attacks, namely the False Data Injection Attacks (FDIAs), targeting state estimation in electric power grids. The first one to come up with FDIAs was Liu *et al.* in [24], who showed that there exist specific scenarios where carefully constructed structured attacks can bypass the traditional BDD. The authors focused on constructing and testing FDIAs on state estimation using the DC power flow model. In general, research works in this area are divided upon the DC and AC power flow model, whereby some authors focus only on one area and others on both. The differences in assumption regarding those two power flow models were explained earlier in Section 2.3.

Although there already existed a variety of methods to detect and identify bad data, Liu *et al.* [24] were able to construct an attack that bypasses all of those approaches. Despite the variation in implementation of bad data detection and identification methods, they all rely on the same mathematical concept: comparing the normalized measurement residuals to a specific threshold, i.e., $||z - H\hat{x}|| > \lambda$. This property can be exploited in a way to bypass all approaches based on this method. Liu *et al.* grouped FDIAs into the following two attack goals and attack scenarios:

$G_1$ **Goal I - Random false data injection attacks.** Using random FDIAs, the attacker tries to find any attack vector that leads to a wrong state estimation. This one is usually easier to construct.

$G_2$ **Goal II - Targeted false data injection attacks.** When constructing targeted FDIAs, the attacker aims to find an attack vector that introduces an error into specific (targeted) state estimation variables. Although this attack is harder to construct, it might potentially cause more damage.

$S_1$ **Scenario I - Limited access to meters.** In this scenario, the attacker is limited to a specific amount of meters. This might be due to physical constraints (e.g., video camera surveillance) or other means of protection. Meters in substations may be harder to access than meters of end-consumers.

$S_2$ **Scenario II - Limited resources available to compromise meters.** The attacker is limited in its resources (e.g., computational power) to compromise only $k$ out of $m$ meters. In this scenario, the attacker may also be interested in finding the minimum number of meters that need to be compromised, given the limited resources available.

Next, the basic principle of constructing FDIAs is outlined, whereas the specific construction of FDIAs with regard to the previously listed goals and scenarios is omitted

for brevity, but can be found in [24]. Different types of attacks with regard to certain constraints, objectives and applications will be discussed in Chapter 3.

Let $z = (z_1, \ldots, z_m)^T$ be the vector of original measurements and $z_a = (z_{a1}, \ldots, z_{am})^T$ the vector of observed measurements that may contain malicious data. The vector of observed measurements can be constructed by adding an attack vector $a = (a_1, \ldots, a_m)^T$ to the vector of original measurements as follows: $z_a = z + a$. In order to construct an attack vector that can bypass the traditional bad measurement detection, the attacker must know the power grid topology and must know the measurement matrix $H$. With regard to Liu *et al.* in [24] the attack vector can then be constructed according to the following theorem:

> *THEOREM 3.1 [24]. Suppose the original measurements $z$ can pass the bad measurement detection. The malicious measurements $z_a = z + a$ can pass the bad measurement detection if $a$ is a linear combination of the column vectors of $H$, that is, $a = Hc$.*

Let the malicious vector of state variables $\hat{x}_{bad}$ be as follows:

$$\begin{aligned} \hat{x}_{bad} &= (H^T W H)^{-1} H^T W z_a = (H^T W H)^{-1} H^T W (z + a) \\ &= \hat{x} + (H^T W H)^{-1} H^T W a \end{aligned} \tag{2.18}$$

Based on THEOREM 3.1, it can then be proven that the $l_2$-Norm of the measurement residuals of the original measurements is the same as the $l_2$-Norm of the measurement residuals of the malicious measurements, if $a = Hc$, where $\lambda$ is the detection threshold (proof omitted for brevity):

$$||z_a - H\hat{x}_{bad} = ||z - H\hat{x}|| \leq \lambda \tag{2.19}$$

The nonzero vector $c$ reflects the error that is introduced to the state estimation variables $\hat{x}_{bad}$, that is $\hat{x}_{bad} - \hat{x} = c$.

This basic method of constructing FDIAs is also used in the practical part of this thesis. Thus, for better understanding, an example using real values built on the already defined 3 bus system depicted in Figure 2.4, is given. For clarity, the variables are summarized again below.

Let the vector of original measurements, as specified in Section 2.3.2, be $z = [0.62, 0.06, 0.37]^T$. The measurement matrix $H$ and the matrix of the corresponding meter errors $W$ is given as follows:

$$H = \begin{bmatrix} 2.5 & -2.5 \\ 1.25 & 0 \\ 0 & -2 \end{bmatrix}, \qquad W = \begin{bmatrix} 10000 & & \\ & 10000 & \\ & & 10000 \end{bmatrix}$$

Performing DC state estimation, this results in the state estimation vector $\hat{x} = [0.0571, -0.1886]^T$. The state estimation variables represent the phase angles at buses 1 and 2 in radians. Keep in mind that the full topology and the matrices $H$ and $W$ are known to the attacker and that the attacker can tamper meter measurement from all $m$ meters.

The attacker now wants to introduce an error of 0.2 radians to $\theta_1$ and 0.4 radians to $\theta_2$, thus $c = [0.2, 0.4]^T$. Next, the attack vector $a$ can be constructed, which is then added to the set of original measurements $z$, resulting in the set of malicious measurements $z_a$.

$$a = Hc = \begin{bmatrix} 2.5 & -2.5 \\ 1.25 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.25 \\ -0.8 \end{bmatrix}$$

$$z_a = z + a = \begin{bmatrix} 0.62 \\ 0.06 \\ 0.37 \end{bmatrix} + \begin{bmatrix} -0.5 \\ 0.25 \\ -0.8 \end{bmatrix} = \begin{bmatrix} 0.12 \\ 0.31 \\ -0.43 \end{bmatrix}$$

Below, the result of DC state estimation according to Equation 2.18 using the malicious set of measurements is shown. As can be seen, subtracting the original state estimation vector $\hat{x}$ from the malicious state estimation vector $\hat{x}_{bad}$ gives the vector of errors $c$.

$$\hat{x}_{bad} = \begin{bmatrix} 0.2571 \\ 0.2114 \end{bmatrix}, \qquad c = \hat{x}_{bad} - \hat{x} = \begin{bmatrix} 0.2571 \\ 0.2114 \end{bmatrix} - \begin{bmatrix} 0.0571 \\ -0.1886 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$

As can furthermore be seen, the values of $z_a$ are different from the values of $z$. However, calculating the $l_2$-norms as specified in Equation 2.19, gives the same result for both the correct measurements as well as the attacked:

$$||z_a - H\hat{x}_{bad}|| = 0.0146$$
$$||z - H\hat{x}|| = 0.0146$$

This shows that the traditional BDD methods described in Section 2.3.3 are not able to detect the malicious measurements and the FDIA could be executed successfully.

Based on these findings, additional approaches were considered and developed by several authors who tried to tackle this shortcoming in traditional BDDs. Chapter 3 will give an overview on the state-of-the-art and investigate the works in more detail.

# Chapter 3

# Related Work

In the previous chapter, state estimation and its importance for the modern power grid was explained. Moreover, the concept of bad data processing was discussed and it was shown that there exists a specific type of attack, the False Data Injection Attacks (FDIAs), that can bypass the traditional Bad Data Detector (BDD).

Although practical implementations in this work mainly deal with the detection of FDIAs, this chapter gives a thorough investigation of related work in the areas of FDIA construction, detection and prevention, because all of these areas are being actively researched at the time of writing and proposed approaches often overlap between these areas. A special focus is on methods that utilize machine learning or deep learning in order to detect FDIAs. Furthermore, limitations of current detection methods are discussed. This chapter concludes by discussing the scope of this work and the improvements over the state of the art with regard to the detection of FDIAs.

Tables 3.1, 3.2 and 3.3 give an extensive overview of all related work, sorted by the year of publication, that has been considered and investigated by the author of this thesis. A division upon distinct categories is attempted to the author's best efforts by utilizing the already existing classification works of He *et al.* [44] and Liang *et al.* [45]. The findings in those classification works are enhanced according to the current state of the art at the time of writing and changes in classification are made. It has to be noted that a strict classification into research works that investigate either construction, detection or prevention of FDIAs is difficult, because many works cover all of the three areas in one research paper. Some authors may start by considering a specific attack scenario, for example a subset of measurements that is already protected, then continue with formulating an attack that is also constructible in case of that specific scenario

and conclude with an improved detection or protection scheme or both. Many works furthermore contain only slight improvements over already existing methods, leading to a huge amount of different works available at the time of writing. When several works describe the same concept with only slight differences between one another, all works are referenced, but the main concept is described only once. Some works may appear in only one category while others are classified into several. In general, the main contribution of a work is used in order to determine the corresponding category or categories.

## 3.1 Attacks on State Estimation

Before going into the details of detection and protection schemes, different types of attacks have to be discussed. As mentioned by Liang *et al.* [45], if an attacker knows the entire system topology, it is straightforward to construct the attack vector $a = Hc$ and launch a successful FDIA. In practice, however, the attacker is usually limited due to certain constraints, incomplete information or other shortcomings. Thus, as can be seen in Table 3.1, attacks on state estimation can furthermore be divided into three major categories: the construction of valid FDIAs, application researches on the impacts of attacks on state estimation and the construction of different types of attacks against state estimation.

### 3.1.1 Construction of Valid FDIAs

In their preliminary version from 2009, Liu *et al.* [51] were the first to introduce FDIAs on DC state estimation under the assumption that the attacker has gained full access to the power system configuration. In 2011, they extended their work to the full version [24], that, since then, triggered numerous researchers to investigate the construction of different FDIAs as well as protection and detection methods.

#### Under the AC Power Flow Model

Rahman *et al.* [48] were one of the first to investigate the construction of FDIAs specifically targeting the nonlinear AC power flow model. They motivated their research by stating that AC state estimation is widely used in the power industry and therefore equally important, if not more important, than the simplified version of DC state

Table 3.1 Related work on attacks on state estimation.

| Research | Categories | Related Works |
|---|---|---|
| Construction of valid FDIAs | Under the AC power flow model | Hug *et al.*, 2012 [46]; Jia *et al.*, 2012 [47]; Rahman *et al.*, 2013 [48]; Liang *et al.*, 2016 [49]; Liu *et al.*, 2017 [50] |
| | Under certain constraints | Liu *et al.*, 2009 [51]; Bobba *et al.*, 2010 [52]; Dan *et al.*, 2010 [53]; Sandberg *et al.*, 2010 [54]; Kim *et al.*, 2011 [55]; Kosut *et al.*, 2011 [56]; Liu *et al.*, 2011 [24]; Sou *et al.*, 2011 [57]; Ozay *et al.*, 2013 [58]; Deka *et al.*, 2014 [59]; Deka *et al.*, 2014 [60]; Hendrickx *et al.*, 2014 [61]; Liu *et al.*, 2014 [62]; Yang *et al.*, 2014 [63]; Hao *et al.*, 2015 [64]; Liang *et al.*, 2016 [49]; Deng *et al.*, 2017 [65]; Yang *et al.*, 2017 [66] |
| | With incomplete information about topology | Esmalifalak *et al.*, 2011 [67]; Rahman *et al.*, 2012 [68]; Kekatos *et al.*, 2014 [69]; Liu *et al.*, 2015 [70]; Yu *et al.*, 2015 [71]; Liu *et al.*, 2017 [50] |
| | With topology being falsified | Kim *et al.*, 2013 [72]; Jia *et al.*, 2014 [73]; Rahman *et al.*, 2014 [74] |
| | Based on PMU measurements | Deng *et al.*, 2017 [75] |
| Application researches on the impacts of attacks on state estimation | Load redistribution attacks | Yuan *et al.*, 2011 [76]; Yuan *et al.*, 2012 [77] |
| | Economic attacks | Xie *et al.*, 2010 [78]; Xie *et al.*, 2011 [79]; Jia *et al.*, 2011 [80]; Choi *et al.*, 2013 [81]; Esmalifalak *et al.*, 2013 [82]; Jia *et al.*, 2014 [73]; Rahman *et al.*, 2014 [74]; Yan *et al.*, 2016 [83]; Kang *et al.*, 2018 [84] |
| | Attacks on contingency analysis | Kang *et al.*, 2018 [84] |
| Different types of attacks | Detectable attacks | Kim *et al.*, 2014 [85]; Gul *et al.*, 2017 [86] |
| | Jamming attacks | Deka *et al.*, 2015 [87]; Deka *et al.*, 2016 [88] |
| | Game-theoretic approaches | Esmalifalak *et al.*, 2013 [82]; Wei *et al.*, 2018 [89] |

estimation. In addition to the construction of FDIAs in DC state estimation, not only offline data about the grid topology is needed, but also online data, which makes the construction of the attack more complex. The authors considered attacks for two scenarios, where the attacker has either perfect or imperfect knowledge about the states of the system. Hug *et al.* [46] proposed a method for constructing FDIAs. Their method, however, requires the attacker to also know the estimated values of specific state variables, which is information that is not easy to obtain. In [49], Liang *et al.* also proposed an FDIA on AC state estimation and studied the impact on both AC as well as DC state estimation by taking into account the physical consequences of FDIAs.

### Under Certain Constraints

Finding sparse attack vectors is motivated by the fact that the attacker only wants to manipulate as few measurements as possible in order to successfully launch an FDIA. By minimizing the number of attacked meters, the risks and costs can be greatly reduced. In other words, minimizing the number of attacked meters means finding an attack vector $a$ that has the minimum number of nonzero elements. As shown by Yang *et al.* [63], however, the problem of finding the least-effort attack for an arbitrary measurement matrix $H$ can be transformed to the *Minimum Subadditive Join problem*, which is NP-hard. Thus, various solutions have been proposed based on different concepts.

Although finding the sparsest unobservable data attack problem is NP-hard in general, Hendrickx *et al.* [61] stated that they have been the first to come up with a polynomial time solution for the case when the network is completely measured. In [24], Liu *et al.* developed a scheme based on heuristics. Although it is fast when the matrix $H$ is sparse, it can be very slow for a more general matrix $H$. Another downside of this approach is, that it cannot guarantee the construction of $a$ even if it exists and it cannot guarantee the construction of the sparsest attack vector. Kim *et al.* [55] proposed a direct $l_1$ relaxation approach based on optimizing the $l_1$ norm, which is known to promote sparsity. One disadvantage, however, is that the solution may not be the sparsest attack vector, because of the relaxation from $l_0$ to $l_1$. Kosut *et al.* [56] developed the notion of a *strong attack regime* and a *weak attack regime*. The first is given, when the attacker has access to a sufficient number of meters to launch an unobservable attack. In the latter, the attacker does not have access to enough meters and the attacks can be detected. By strictly applying graph theory, the authors developed a polynomial time algorithm to determine the minimum number of tampered meters in order to launch an

unobservable attack. Ozay *et al.* [58] and Liu *et al.* [62] developed attacks by exploiting the sparse structure of the system. Several authors, such as Deka *et al.* [59, 60], also developed algorithms to find the sparsest attack vector by formulating the problem as an optimization problem under the assumption that a few of the measurements are protected. In [64], Hao *et al.* developed a greedy search algorithm in order to find the sparsest attack vector. In one of the more recent works from 2017, Yang *et al.* [66] proposed an effective greedy algorithm based on the reduced row echelon form and demonstrated the efficiency of the algorithm.

### With Incomplete Information About Topology

In contrast to the case, where the entire system topology is known, Esmalifalak *et al.* [67] proposed a method for constructing FDIAs without prior knowledge of the power grid topology based on linear independent component analysis. By just observing the power flow measurements, they were able to estimate both the system topology as well as the power states and utilize that information to construct malicious attacks that do not trigger the traditional BDD. Additionally, instead of concentrating on getting the entire network information, Liu *et al.* [70] proposed an attack method that focuses on a local region. They showed that no information about the non-attacked region is needed.

By collecting both offline and online data, Rahman *et al.* [68] mathematically characterized FDIAs from both the attacker's, as well as the grid operator's point of view. According to them, they were the first to investigate the construction of FDIAs with line admittance uncertainty. In [69], Kekatos *et al.* exploited the correlation between Locational Marginal Prices (LMPs) and the economic dispatch problem, in order to reconstruct the grid matrix from the LMPs. The LMP is used to reflect the electricity price at a particular node or location. A very similar approach of constructing FDIAs without knowing the power grid topology, the so-called blind FDIAs, were obtained by Yu *et al.* [71], where they utilized principal component analysis approximation methods in order to compose the attacks.

### With Topology Being Falsified

In contrast to the manipulation of state estimation, Kim *et al.* [72] were one of the first to investigate attacks on the network topology of a smart grid. By launching a topology attack, the attacker tries to bypass both the bad data and topology error detectors in order to falsify the estimated topology in the control center. If successful, this can lead

to serious consequences. A grid under stress, for example, may appear as normal or a normal grid may appear under stress, causing unnecessary load shedding and other costly actions. The authors furthermore investigated impacts on both the AC and the DC power flow model.

#### Based on PMU Measurements

In [75], Deng *et al.* distinguished between physical attacks and cyber attacks. Physical attacks directly target the power system components in order to cause power outages or trigger cascading failures whereas cyber attacks include the already known FDIAs that target the SCADA/EMS system. When speaking of Coordinated Cyber-Physical Attacks (CCPAs), cyber attacks are utilized in order to mask or hide physical attacks. Although the increased installation of PMUs improves situation awareness for the grid operator, it also makes the grid more vulnerable to attacks. In their work, the authors investigated the construction of CCPAs based on PMU measurements, developed two different FDIAs to mask physical attacks and proposed corresponding countermeasures.

### 3.1.2 Application-Specific Research

In addition to the construction of valid FDIAs, several authors investigated the impacts of attacks on state estimation such as load redistribution attacks, economic attacks and attacks on contingency analysis.

#### Load Redistribution Attacks

Load Redistribution (LR) attacks, a special type of FDIAs, are attacks where only load bus injection measurements and line power flow measurements are attacked in order to mislead the Security-Constrained Economic Dispatch (SCED) module. Yuan *et al.* [76] were the first to come up with this type of attacks. In [76] and [77] they investigated the impact of LR attacks on two different attacking goals: the immediate attacking goal and the delayed attacking goal. They furthermore developed the most damaging LR attack and proposed a corresponding protection scheme. Their proposed LR attack can lead to immediate load shedding or even delayed load shedding, potentially physically damaging the power system.

#### Economic Attacks

Another incentive for an attacker is to generate profit by manipulating electricity prices and electricity pricing. In 2010 and 2011, Xie *et al.* in [78] and Xie *et al.* [78] examined the impact of FDIAs on electricity markets and pricing. The main objective of the attacker is to buy virtual power at a lower price, manipulate the LMPs through FDIAs and then sell virtual power at a higher price. They showed that by combining FDIAs and virtual bidding, financial profit can be generated for the attacker. Jia *et al.* [80] proposed an optimal attack strategy for influencing revenues in electricity markets by considering congestion patterns which affect the LMP the most. In [81], Choi *et al.* investigated the impact of system topology errors on real-time electricity market prices. Due to the fact that the Optimal Power Flow (OPF) routine relies on both the topology processor and the state estimation, Rahman *et al.* [74] investigated the impact of topology attacks on the OPF routine and subsequently the economic operation. Yan *et al.* [83] examined the impact of FDIAs in the case of blackouts and analyzed and simulated these scenarios from a high-level point of view.

#### Attacks on Contingency Analysis

In one of the more recent works from 2018, Kang *et al.* [84] considered a new class of FDIAs on contingency analysis through state estimation. They showed that by manipulating contingency pairs of transmission line flows, the LMPs of real-time power markets can be manipulated.

### 3.1.3 Different Types of Attacks

In addition to FDIAs, usually also referred to as *stealthy* attacks or *hidden* attacks, there exist additional types of attacks that affect state estimation despite triggering the BDD. Moreover, several works investigated game-theoretical approaches in order to model both attack and defense of FDIAs.

#### Detectable Attacks

*Detectable* data attacks or *data integrity* attacks work by initially failing the bad data detection, but succeeding once the BDD removed the bad data. In [85], Kim *et al.* constructed a detectable attack where they showed that when focusing on the bad data

identifier, the cardinality of the detectable attacks can be reduced by more than 50% compared to the cardinality of hidden attacks. They furthermore showed that, although studying the attack based on the DC power flow model, the state estimate was also perturbed when launching the attack against the nonlinear AC power flow model. Gul *et al.* [86] furthermore enhanced the aforementioned approach by designing attacks that solely consist on re-ordering of the measurement vector, given two different scenarios based on AC state estimation.

### Jamming Attacks

Another concept of manipulating the state estimate is to prevent the state estimator from receiving a particular measurement. This adversarial action is known as *jamming* and has been investigated by Deka *et al.* [88]. In [87], they stated that detectable attacks, in comparison to stealthy attacks, have lower costs and a wider feasible operation region. They furthermore developed a polynomial time approximate algorithm for the construction of the attack vector. In [88], they investigated jamming in the context of both generalized stealthy data attacks as well as generalized detectable data attacks. When developing the framework, the authors considered an attacker capable of performing the following three actions: 1) jamming; 2) injection of data in insecure measurements; and 3) jamming of secure measurements.

### Game-Theoretic Studies

Additionally, several works applied game theoretical approaches to model both the construction of FDIAs as well as the defense against FDIAs. Esmalifalak *et al.* [82] modeled attacking and defending the measurements as a zero-sum game between the attacker and the defender and investigated the impact on electricity prices. They also showed that the attacker is specifically able to lower the prices. Wei *et al.* [89] proposed a stochastic game model that simulates the interactions between a malicious attacker and a defender.

In response to all these different kinds of attacks and threats, lots of efforts have been made to develop efficient countermeasures. As mentioned by Chaojun *et al.* [90], research can be specifically classified into the following two categories: i) protection-based and ii) detection-based. In the next two sections, both categories are investigated in more detail.

Table 3.2 Related work on protection against attacks on state estimation.

| Categories | Related Works |
|---|---|
| Protect a set of basic measurements | Bobba *et al.*, 2010 [52]; Dan *et al.*, 2010 [53]; Bi *et al.*, 2011 [91]; Kim *et al.*, 2011 [55]; Hug *et al.*, 2012 [46]; Bi *et al.*, 2014 [92]; Deka *et al.*, 2014 [59]; Deka *et al.*, 2014 [60]; Yang *et al.*, 2014 [63]; Anwar *et al.*, 2015 [93]; Hao *et al.*, 2015 [64]; Wickramaarachchi *et al.*, 2016 [94] |
| PMU-based protection methods | Kim *et al.*, 2011 [55]; Deng *et al.*, 2017 [75]; Yang *et al.*, 2017 [66] |
| Other protection methods and applications | Hug *et al.*, 2012 [46]; Talebi *et al.*, 2012 [95]; Deng *et al.*, 2017 [65] |

## 3.2 Protection-Based Approaches

In this section, protection-based approaches are investigated for both AC as well as DC state estimation. A more general overview is found in Table 3.2.

### 3.2.1 Protect a Set of Basic Measurements

In 2010, Sandberg *et al.* [54] proposed two security indices for state estimators in order to provide the power grid operator with a tool to identify sparse data manipulation patterns by locating power flows whose measurements are easy to manipulate. Based on those security indices, Dan *et al.* [53] developed three algorithms to obtain both perfect and partial protection against FDIAs with respect to a limited budget for protection. Bobba *et al.* [52] developed an algorithm based on brute-force search, to find a basic set of measurements that needs to be protected such that no undetectable FDIA can be launched. They furthermore state that the number of measurements that needs to be protected is the same as the number of unknown state variables in the state estimation problem. Because this could lead up to several hundred in a large scale power system, this would end up being a costly task. When Milosevic *et al.* [96] and Esmin *et al.* [97] determined that it is sufficient to only protect the part of state estimates that are considered to be critical to maintain normal system operations, Bi *et al.* [91] developed an algorithm that determined the number of measurements that need to be protected in

order to secure only this specific subset of state estimation variables. Consequentially, not all measurements had to be protected but only the subset. Later on, Kim *et al.* [55] developed a generalized framework to construct sparse FDIAs when only a subset of measurements is protected. Based on that framework they proposed an algorithm that identifies key measurements in this subset, that, when protected, lead to an increase in the minimum number of meters that the attacker has to compromise, thus increasing the difficulty for the attacker.

Next, several authors proposed graph-theoretic approaches in order to find a set of measurements that needs to be protected to make the construction of FDIAs unfeasible. Deka *et al.* [59, 60] proposed a graph-theoretic approach based on the max-flow min-cut theorem to determine the optimum number of measurements that need to be tampered to make an FDIA possible. Furthermore, they discussed a polynomial-time solvable algorithm to solve this problem under different conditions. Bi *et al.* [92] carefully selected meter measurements that need to be protected to make an FDIA unfeasible by solving a variant Steiner tree problem in a graph. In [94], Wickramaarachchi *et al.* utilized the approach of Bi *et al.* [92] to show that the protection scheme of Deka *et al.* in [59] and [60] does not hold true in every scenario. They identified a class of attack vectors, where attacks, constructed from this class of attack vectors, cannot be defended against using the protection scheme from Deka *et al.* Following, they proposed an enhanced protection scheme based on a minimum Steiner tree. Hao *et al.* [64] also developed a graph-theoretic protection scheme and proposed an algorithm to efficiently construct and detect highly sparse undetectable attack vectors. Yang *et al.* [63] considered the scenario where the attacker wants to find a set of meters that, when compromised, cause maximum damage. Based on that, they then proposed a protection-based defense scheme that identifies the critical sensors that need to be protected in order to defeat such attacks.

### 3.2.2 PMU-based Protection Methods

One disadvantage of PMUs is the high capital cost, making a large scale deployment difficult. Thus, several authors investigated the optimal placement of PMUs as well as the impact of using PMUs for protection in general. In [55], Kim *et al.* developed a greedy algorithm for determining buses on which PMUs should be placed, such that the total number of PMUs is minimized. Deng *et al.* [75] investigated the impact of PMUs on the feasibility of FDIAs in general. In one of the more recent works from 2017, Yang *et al.* [66] developed a greedy-based algorithm for determining the optimal PMU

placement to defend data integrity attacks while maintaining system observability. The authors hereby improved the work of Kim *et al.* [55] by enhancing their own method from 2014 [63].

### 3.2.3 Other Protection Methods and Applications

Talebi *et al.* [95] proposed a dynamic reconfiguration of the power grid into several microgrids to make it impossible for the attacker to launch a synchronized FDIA . Hug *et al.* [46] investigated protection against FDIAs in the context of AC state estimation and examined the differences between protecting AC and DC state estimation. Moreover, they introduced analytical techniques for vulnerability analysis of state estimation. In [65], Deng *et al.* addressed the problem of financial cost that is associated with ultimately protecting a set of smart meters. They proposed an algorithm to find a least-budget defense strategy.

## 3.3 Detection-Based Approaches

In addition to protection-based methods, many authors investigated methods in order to detect attacks on state estimation. This mitigates two drawbacks of protection-based methods. First, redundancy is not reduced, because all measurements can be used instead of only the secured and protected ones. Second, because of the fact that protection cannot be secure 100% of the time, attacks can also be detected in cases where the protection-based approaches fail. As shown in Table 3.3, detection methods are classified into three categories, whereby major attention is drawn towards machine learning-based approaches.

### 3.3.1 General Detection Methods

One of the first authors proposing a detection-based defense scheme was Kosut *et al.* [98] in 2010. The authors proposed a detector based on the Generalized Likelihood Ratio Test (GLRT) to detect FDIAs even when the BDD was failing. Their proposed detector also tries to identify the meters that originated the attack. They furthermore showed, that conventional bad data can clearly be separated from bad data occuring from a malicious attack. Liu *et al.* [62] interpreted the detection problem as a matrix separation problem and proposed an approach that separates nominal power grid states

Table 3.3 Related work on detection of attacks on state estimation.

| **Research** | **Related Works** |
|---|---|
| General detection methods | Kosut *et al.*, 2010 [98]; Huang *et al.*, 2011 [99]; Bhattarai *et al.*, 2012 [100]; Liu *et al.*, 2013 [101]; Liu *et al.*, 2014 [62]; Manandhar *et al.*, 2014 [102]; Yang *et al.*, 2014 [63]; Chaojun *et al.*, 2015 [90]; Chen *et al.*, 2015 [103]; Hao *et al.*, 2015 [64]; Li *et al.*, 2015 [104]; Yu *et al.*, 2015 [105] |
| Distributed detection methods | Pasqualetti *et al.*, 2011 [106]; Talebi *et al.*, 2012 [95]; Sedghi *et al.*, 2013 [107] |
| ML-based detection methods | Ozay *et al.*, 2012 [108]; Chakhchoukh *et al.*, 2016 [109]; Yan *et al.*, 2016 [110]; Esmalifalak *et al.*, 2017 [111]; Foroutan *et al.*, 2017 [112]; He *et al.*, 2017 [113]; Wang *et al.*, 2017 [114]; Ayad *et al.*, 2018, in press [115]; Wang *et al.*, 2018 [116] |

from anomal ones. One downside of these approaches is, that they are not able to detect bad data in measurements that fit the distribution of historical measurements. In response to this, Chaojun *et al.* [90] addressed this issue by proposing a detector based on the Kullback-Leibler Distance (KLD), which calculates the distance between two probability distributions and is therefore able to detect false data that comes from the same distribution.

Huang *et al.* [99] proposed an adaptive cumulative sum (CUSUM) algorithm that is able to perform sequential detection, also known as *Quickest Detection (QD)*. Instead of using a fixed-sample size, sequential detection operates on sequential data, thus being more suitable for real-time monitoring. In order to use the CUSUM algorithm, the probability distribution before and after the attack has to be known. To fulfill this requirement, the authors assumed a Gaussian distribution with fixed mean and covariance for the state vector prior to the attack and also assumed that the malicious data is small and positive in magnitude. This makes the detection method unsuited for large attacking data. In [104], Li *et al.* proposed an improved CUSUM-type algorithm based on the Generalized Likelihood Ratio (GLR) that is robust to arbitrary state variables and arbitrarily injected data.

Hao *et al.* [64] introduced a detection method based on the principal component analysis problem, which is shown to identify both real measurements and attacks also in the case, where only partial observations can be collected due to noise. In [102], Manandhar *et al.* improved the traditional $\chi^2$ test based BDD by proposing an euclidean detector that

continuously monitors the difference between the estimated values and the measured values and thus detects if an FDIA is present, or not.

Bhattarai *et al.* [100] and Yu *et al.* [105] added a secure watermark to real-time measurement readings to make it possible for the utility to detect the presence of an adversary. Bhattarai *et al.* stated that this technique can effectively detect any false manipulation of the measurement readings at low cost.

### 3.3.2 Distributed Detection Methods

In [106], Pasqualetti *et al.* proposed two distributed methods for state estimation and FDIA detection. They assumed that multiple control centers are coordinating between each other. Talebi *et al.* [95] partitioned the grid into a group of microgrids, which are coordinating between each other and are thus able to detect coordinated attacks under specific circumstances. Liu *et al.* [101] proposed a detection scheme named adaptive partitioning state estimation, which divides the large system into several smaller subsystems in order to improve the sensitivity for bad data. They furthermore showed that their detection scheme is applicable to AC state estimation. Sedghi *et al.* [107] proposed a decentralized detection scheme based on a Markov graph of bus phase angles. They stated that their detection scheme is successful regardless of the size of the attacked subset.

### 3.3.3 Machine Learning-Based Detection Methods

Influenced by the increasing popularity of machine learning in recent years, several authors utilized machine learning techniques to detect attacks on state estimation. This section gives an overview of the state of the art and emphasizes the importance of not only constructing the models but also evaluating the models. Equally important is the generation of attacks and data, which should be documented and designed carefully. A quick overview of all related and investigated works can be found in Table 3.4. As to the author's best knowledge, there is no public real-world data set containing FDIAs available. Fortunately, the simulation of different types of attacks according to the literature in Section 3.1, is possible.

In [111], Esmalifalak *et al.* proposed two machine learning algorithms to detect stealth attacks on DC state estimation. The first method utilizes supervised learning to train a distributed SVM with a Gaussian kernel whereas the second approach takes

Table 3.4 Detailed classification of machine learning-based detection methods

| Related Work | AC / DC | Proposed Methods | Compared Methods | Metrics | Test-case |
|---|---|---|---|---|---|
| Ozay *et al.*, 2012 [108] | DC | kNN, SVM, SLR | SVE | Acc, Prec, Rec, $F_1$ | 9, 30, 57, 118 |
| Chakhchoukh *et al.*, 2016 [109] | AC | DRE | SVM, AD | Rec | 118 |
| Ozay *et al.*, 2016 [117] | DC | Perceptron, kNN, SVM, SLR, S3VM, decision and feature level fusion algorithms | | Rec | 118 |
| Yan *et al.*, 2016 [110] | AC | SVM, kNN, ENN | | Acc, Prec, Rec | 9, 57, 118 |
| Esmalifalak *et al.*, 2017 [111] | DC | Distributed SVM, AD | | $F_1$ | 118 |
| Foroutan *et al.*, 2017 [112] | DC | Mixture Gaussian Distribution | SVM, MLP, AD | $F_1$, ROC | 118 |
| He *et al.*, 2017 [113] | DC | CDBN architecture | SVM, ANN | Acc, ROC | 118, 300 |
| Wang *et al.*, 2017 [114] | AC | Margin Setting Algorithm | | Acc | 6 |
| Ayad *et al.*, 2018, in press [115] | DC | RNN | | Acc, Prec, Sensitivity, Specificity | 30 |
| Wang *et al.*, 2018 [116] | AC | Deep learning based ISE | Persistence method, BP, Shallow SVM | Acc, MAPE | 9, 14, 30, 188 |
| Yu *et al.*, 2018 [118] | AC | Wavelet transform feature extracting, RNN with GRU | KLD, Kalman Filter, Sparse Optimization | Acc, FP, FN | 118, 300 |

advantage of unsupervised learning by classifying points as anomalies that do not obey a fitted multivariate Gaussian distribution. In both cases, the authors applied Principal Component Analysis (PCA) for preprocessing and showed that normal data and tampered data tend to be strictly separated in a projected space. They showed that by only keeping two of the principal components, 99% of the variance could be retained. In their paper they used the $F_1$ score as performance metric and evaluated the model by computing $F_1$ on the cross-validation set. In addition to the unsupervised

method, they proposed a semi-supervised approach, where the output labels are used to learn the best threshold for the unsupervised method.

Chakhchoukh *et al.* [109] used an unsupervised machine learning technique called *density ratio estimation* [119] to detect stealthy attacks on AC state estimation. They showed improved performance over the distributed SVM and the Gaussian model from Esmalifalak *et al.* [111]. As performance metric, recall was used.

Ozay *et al.* [108] investigated three supervised learning algorithms which observe the power system in order to construct a training data set that is then used to detect attacks on future data. They modeled the detection problem as a supervised binary classification problem and used k-nearest Neighbor (kNN), SVM and Sparse Logistic Regression (SLR) algorithms to detect the attacks. Ozay *et al.* [117] extended the aforementioned work by conducting a detailed analysis of the three supervised learning algorithms and additionally proposed semi-supervised and online learning settings for kNN, SVM and SLR. They furthermore examined decision- and feature-level fusion algorithms and showed that these types of algorithms are more robust to changes in system size and data sparsity. The authors evaluated all algorithms on simulations and used precision, recall and $F_1$ as performance metrics.

Yan *et al.* [110] also investigated supervised learning based detectors based on the following algorithms: SVM, kNN and Extended Nearest Neighbor (ENN). They evaluated their methods on both balanced as well as imbalanced data sets. In comparison to [117], the authors constructed the attack vectors in the state variable space instead of the measurement space. Moreover, in contrast to Yan *et al.*, the authors in [117] only considered attacks with different degrees of sparsity, while the energy of attack vectors was not considered.

Foroutan *et al.* [112] utilized a semi-supervised learning approach based on mixture Gaussian distribution. They evaluated their model on minimum energy residual attacks and sparse attacks [98, 61] and showed superior performance over already existing methods. Chosen approaches for comparison were SVM, Multi-Layer Perceptron (MLP) and anomaly detection based on a multivariate Gaussian distribution.

He *et al.* [113] utilized deep learning for detection of FDIAs by proposing an extended Deep Belief Network (DBN) architecture. They compared their model to an SVM with a Gaussian kernel and an Artificial Neural Network (ANN) with one hidden layer and 25 hidden units and showed that it outperforms both of the compared models.

Wang *et al.* [114] proposed a data analytical method based on data-centric paradigm employing the Margin Setting Algorithm (MSA). By comparing their model to an SVM and an ANN, they showed that they achieve better performance. They furthermore stated that MSA is a relatively new machine learning algorithm and that they have been the first to apply MSA for the detection of FDIAs.

Wang *et al.* [116] devoted their research to the investigation of cyber-attack modeling of AC grids and corresponding defense mechanisms. They applied a deep learning based Stacked Auto-Encoder (SAE) to extract nonlinear and non-stationary features from the electric load data. These features can then be used to improve the accuracy of electric load forecasting, which narrows down the width of intervals of state variables and thus increases the chance for detection. They showed that SAE in combination with their proposed Interval State Estimation (ISE) based defense mechanism is 100% effective against the anomalies introduced by their chosen attacks. They considered both stealthy attacks as well as attacks with incomplete information about topology. It has to be stated, that their attacks did not directly introduce an error to the state estimate, but aimed to overload a given line by manipulating least measurements.

Ayad *et al.* [115] proposed a detection scheme based on RNNs. In their approach, they limited the number of previous output states that are used to predict the current output to five because of gradient explosion. The authors did not make any comparisons to other machine learning-based approaches.

# Chapter 4

# Machine Learning

This chapter starts by outlining the concept of machine learning, as well as different learning schemes and basic terminology. Supervised learning methods such as SVM and NN are investigated in more detail. A special focus is placed on RNNs, which are well suited for data sets that do not only feature spatial dependencies, but also temporal ones. LSTM units are investigated to improve the performance of an RNN for learning long-range dependencies. Different types of preprocessing methods and performance metrics are outlined with a focus on imbalanced classes.

## 4.1 Introduction to Machine Learning

The term *machine learning* can be defined in multiple ways and is closely related to a broad range of fields in computer science, like pattern recognition and artificial intelligence. According to Alpaydin in [120], machine learning is a way of programming computers to optimize a specific performance criterion using example data or data from past experience. For many tasks there exist specific algorithms to find a solution. For example, filtering an array of strings based on a certain pattern. For other problems, however, no unique algorithm or solution can be found. Examples are hand-written digit recognition or classifying whether an email is spam or not. In these cases, given that data is available, knowledge can be extracted from the data and utilized to train machine learning models in order to output solid and useful approximations.

Basic terminology and concepts are explained by referring to the example of hand-written digit recognition, loosely based on the example of Bishop in [121]. The goal is

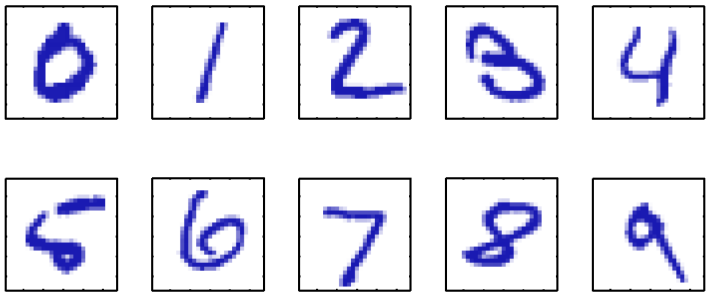


Figure 4.1 Set of hand-written digits [121].

to build a machine that takes a vector $s$ as input and outputs the actual displayed digit $0, \ldots, 9$. The input data consists of $N$ $26 \times 26$ pixel images $s_1, \ldots, s_N$. In Figure 4.1, an example for each of the hand-written digits can be seen.

In addition to the input vector $s$, each sample of the input data includes a *target vector* $y$, containing the actual digit of the image. This type of true information is also known as *ground-truth data*. In order to perform learning and being able to determine the performance of the model after learning has finished, input data is divided into three sets, namely the *train*, *validation* and *test* data set. Training data is used to train the machine learning model, which results in the determination of a set of parameters that allow the best possible classification. After the model has been trained, an image from the validation data is used as input, whereby the model generates an output $\hat{y}(x)$, linking to one of the digits $0, \ldots, 9$. This is repeated for all images in the validation data set. Based on the fact, that both the real target as well as the predicted target of each image of the validation data set are available, performance measurements on how well the model performs on new examples, can be calculated. The term *generalization* refers to how well a model performs on examples, that were not used during training.

This type of learning, where input data consists of both the input vectors as well as the corresponding labels, is denoted as *supervised learning*. In the above case of hand-written digit recognition, where the goal was to assign the input vector to one or more categories, the produced output is used to solve a so-called *classification* problem. Outputs of *regression* problems on the other side consist of one or several continuous variables. An example would be the determination of the price of a house, given the area, the number of residents and the number of rooms. Another type of learning is *unsupervised learning*, where no label is supplied to the input vector and the goal may be to group the data into several different *clusters* based on similarities or dissimilarities.

Depending on the type of raw data, *preprocessing*, also known as *feature extraction*, must be applied, or not. This can be the case when dimensionality has to be reduced, raw

data is noisy or simply consists of different ranges of values (e.g., one feature contains the number of residents in the size of one to ten and another feature represents the area of the house in the size of hundreds of square meters). When preprocessing data, it has to be taken into consideration that no important information is lost.

Another key component in machine learning is *model selection*. Choosing a model that is too simple for the given data can result in a *high bias* problem (underfitting), whereas a highly complex model, that resembles the data perfectly, may result in a *high variance* problem (overfitting). Models with a high variance problem tend to generalize badly to new data. In general, model complexity should be high enough to capture enough information in the data to successfully differentiate between different classes to a high degree, but should be kept as simple as possible. This behaviour is also known as *Occam's razor* [122], who stated that, when in favor of several well performing models, the simplest one should be chosen.

For visualization purposes, consider an input data set $s = (s_1, \ldots, s_N)^T$ of size $N$, where the corresponding labels $y = (y_1, \ldots, y_N)^T$ are sampled from the function $sin(2\pi s)$ including a small level of random noise following a Gaussian distribution. The goal is to train a model using the training data, in order to make correct predictions of the target $\hat{y}$ for new values $s$ of the input variable. A polynomial function of order $M$ is used as a model, where $w = (w_0, \ldots, w_M)$ represents the parameters that need to be trained and $s^j$ the polynomial function:

$$\hat{y}(s, w) = w_0 + w_1 s + w_2 s^2 + \ldots + w_M s^M = \sum_{j=0}^{M} w_j s^j \tag{4.1}$$

Functions, such as the polynomial function, are considered to be linear and are therefore part of the so-called *linear models*. In order to be able to evaluate the performance of the model after each iteration over the training data, an *error function* (i.e., cost or loss function) has to be defined. One commonly used error function in machine learning is the residual sum of squares (RSS):

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{\hat{y}(s_n, w) - y_n\}^2 \tag{4.2}$$

The factor $\frac{1}{2}$ is included for ease of later computation. The function will be zero if, and only if, all of the predicted values $\hat{y}(s_n, w)$ are equal to the target values $y_n$.

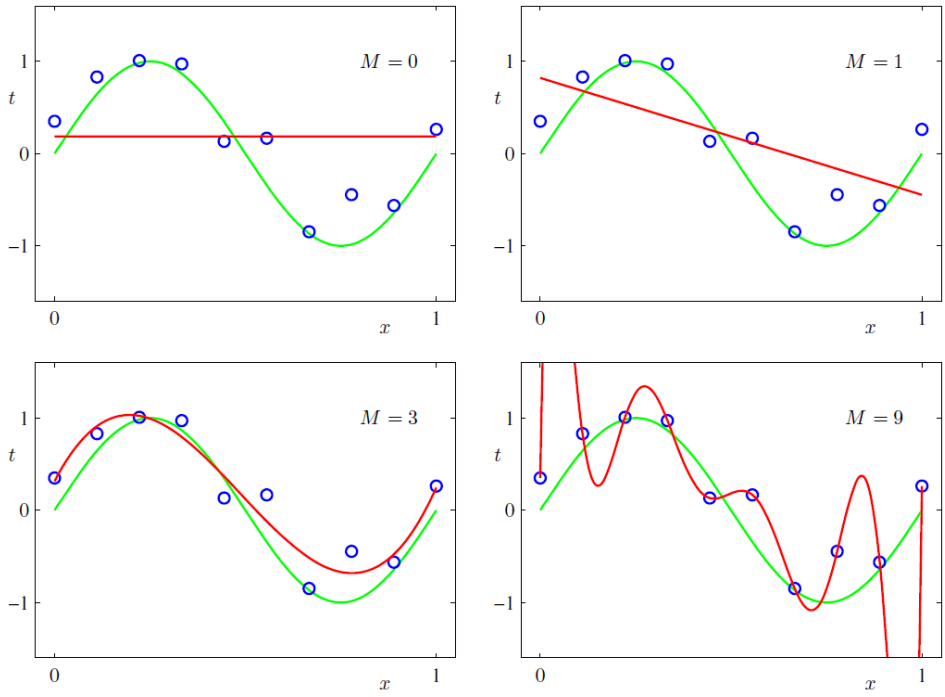



Figure 4.2 Plots of polynomials of varying order $M$ shown as red curves, blue dots representing the input data and the green line displaying the original sine function [121].

The objective of training the model using training data is to find a set of parameters $w$ that minimize the error function. Because the parameters $w$ in Equation 4.1 are linear and the error function is a quadratic function of the parameters, the derivatives of the error function with respect to the parameters are also linear. Thus, a unique solution, denoted by $w^*$, can be found in closed form. A different method of finding solutions would be the use of iterative optimization algorithms such as gradient descent, which will be covered later in Section 4.3 when discussing neural networks. The generalization ability of the model can then be evaluated by making predictions using the test data and calculating the error with regard to the targets of the test data. When it comes to the choice of $M$, the goal is to find a model that is neither causing a high bias nor a high variance problem. An example is given in Figure 4.2, where $M = 3$ tends to be the best representation of the original function, whereas $M = 0$ and $M = 1$ are causing a high bias problem and $M = 9$ is fitting all points perfectly, but is reconstructing the shape of the original sine function badly.

A more general definition of an error function is the root mean square (RMS) function, defined in Equation 4.3, which includes the size of the data set $N$, thus giving error values independent of the size of the data set.

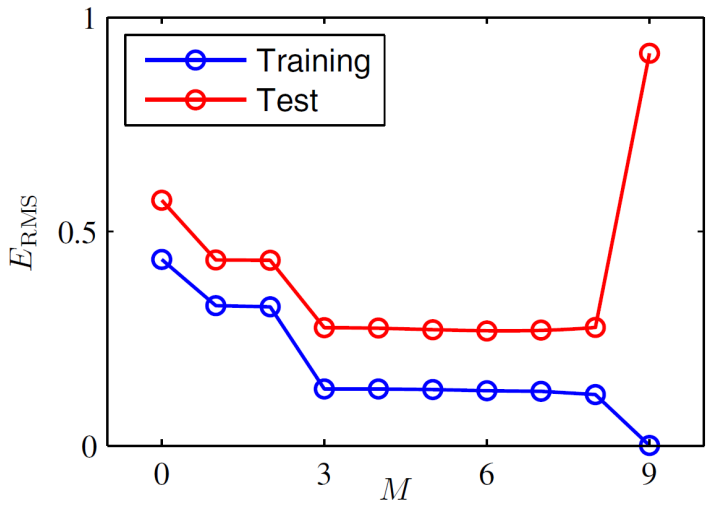



Figure 4.3 Plot of $E_{RMS}$ evaluated on independent test sets for varying values of $M$ [121].

$$E_{RMS}(w) = \sqrt{2E(w)/N} \tag{4.3}$$

As mentioned previously, one way of measuring the generalization ability of a model is to make predictions on the validation data. In Figure 4.3, models with varying polynomial order ($M = 0, \ldots, 10$) were trained using the training data set of size 10. Moreover, the error $E_{RMS}$ was evaluated on a validation data set of size 100 that has been generated with new random noise values in the targets. As can be seen for $M = 9$, the training set error reaches zero because the model is predicting each point correctly, whereas the generalization error becomes very large. This is due to the fact that the model is reproducing the shape of the original sine function badly, as already shown in Figure 4.2.

So far, basic machine learning terminology was defined and the difference between underfitting and overfitting demonstrated by use of a simple model consisting of a $M$ order polynomial function. Next, common concepts to avoid overfitting are investigated.

In the previous example, the training set consisted of ten data points. Thus, the 10-order polynomial model was able to reconstruct the training set perfectly, but achieved a very bad result on data it has not yet seen before. This is due to the fact that the model adjusted better to the Gaussian noise in the data than to the underlying sine function. One way to avoid overfitting is the use of more samples in the training data. An example can be seen in Figure 4.4.

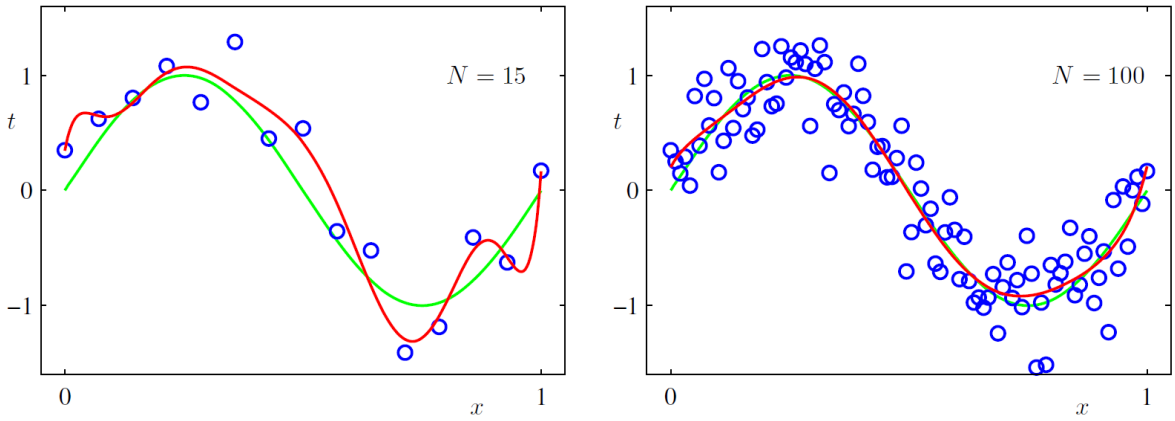



Figure 4.4 Avoid bad generalization by using more data points in the training data set [121].

This approach, however, is not feasible when no additional data can be obtained for a specific problem. In that case, a different technique called *regularization* can be used. Regularization introduces a new parameter $\lambda$ to the error function that introduces a penalty to each of the parameters $w$, preventing them from growing too large. The simplest form of regularization used in an error function is given by the following equation:

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}\Big(\hat{y}(s_n, w) - y_n\Big)^2 + \frac{\lambda}{2}||w||^2 \tag{4.4}$$

where $||w||^2 = w_0^2 + w_1^2 + \ldots + w_M^2$. It has to be noted, that $w_0^2$ is usually excluded from the regularization term in order to not falsify the result. Because the model builds on a polynomial function, the solution to the minimization problem of the error function can again be found in closed form. The impact of regularization on generalization is illustrated in Figure 4.5, where the root mean square error function with different regularization parameters was evaluated for the training and the validation data set.

## 4.2 Support Vector Machines

From a geometrical point of view, the concept of a Support Vector Machine (SVM) can be described by considering the case of a simple binary classification task [123]. Given a training data set $D_{Tr}$, where one sample $s_i$ consists of $m$ features and $Y = \{0, 1\}$, the goal is to find an element $w \in \mathbb{R}^m$ with $||w||_2 = 1$ and a real number $b \in \mathbb{R}$, such that $(w, b)$ forms an affine linear hyperplane that completely separates the samples in the

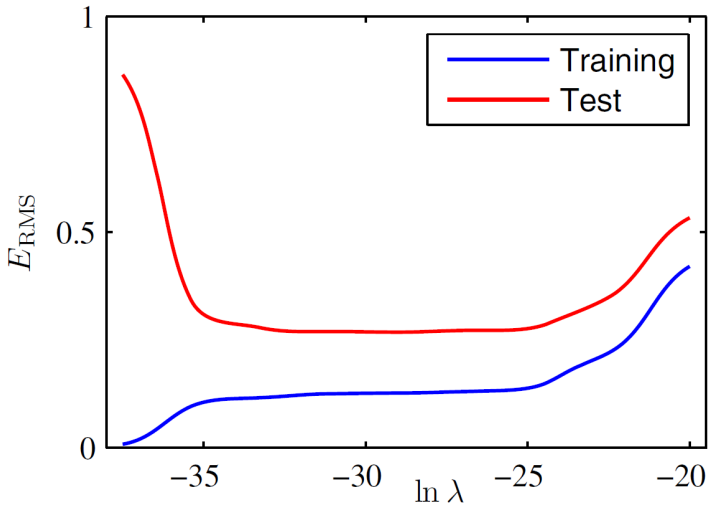



Figure 4.5 Plot of $E_{RMS}$ for various $ln(\lambda)$ using the polynomial model of order $M = 9$ [121].

data set $D$ into the two groups $\{(s_i, y_i) \in D : y_i = 1\}$ and $\{(s_i, y_i) \in D : y_i = 0\}$. One way of finding the maximum margin hyperplane is to minimize $||w||_2$.

If the data set $D$ can not be entirely linearly separated, the input data $S$ can be mapped into a different feature space, also known as Hilbert space $H_0$, by a nonlinear map $\Phi(s_i)$. One disadvantage of this mapping is that the hyperplane ends up in a very high or infinite-dimensional space, making the SVM more prone to overfitting. To counteract this problem, slack variables $\xi_i \geq 0$ can be introduced, resulting in the so-called soft margin SVM. This results in the following quadratic optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \quad \text{for } w \in H_0, b \in \mathbb{R}, \xi \in \mathbb{R}^N \\ \text{s.t.} \quad & y_i(w^T \Phi(s_i) + b) \geq 1 - \xi_i, \; i = 1, \ldots, N \\ & \xi_i \geq 0, \qquad\qquad i = 1, \ldots, N \end{aligned} \tag{4.5}$$

where $C$ is a modifiable positive regularization parameter. Because Equation 4.5 usually needs to be solved in a high or infinite-dimensional space, the Lagrange approach is utilized to convert the problem into the following dual program:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \Phi(s_i)^T \Phi(s_j) \\ & \text{over } \alpha \in [0, C]^N \\ \text{s.t.} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0; \end{aligned} \tag{4.6}$$

where $\alpha$ is the Lagrange multiplier.

This notation allows to replace the inner product $\Phi(s_i)^T \Phi(s_j)$ in Equation 4.6 with any kernel function $k(s_i, s_j)$ that satisfies the following condition:

$$k(s_i, s_j) = \Phi(s_i)^T \Phi(s_j), \quad s_i, s_j \in S. \tag{4.7}$$

In this work, the nonlinear Gaussian radial basis function is used as a kernel for the SVM:

$$k(s_i, s_j) = e^{-\lambda ||s_i - s_j||^2} \tag{4.8}$$

where $\lambda$ is the width of the kernel. Both $C$ and $\lambda$ denote hyperparameters that have to be adjusted accordingly when training the model in order to achieve a well performing SVM.

In the case of multilabel classification, the problem can be viewed as $d$ binary classification problems. This approach is also known as *one-vs-the-rest* classification [121].

## 4.3 Neural Networks

As shown by McCulloch *et al.* [124] in 1943, the nowadays widely used term *neural networks* is motivated by the concept of how information is processed in biological systems.

This section first introduces the *multilayer perceptron*, also known as *feed-forward neural network*, which is one of the most widely used types of neural networks according to Bishop [121]. Following, a variation called the Recurrent Neural Network is investigated, especially suited for modelling sequential data with temporal dependencies. In order

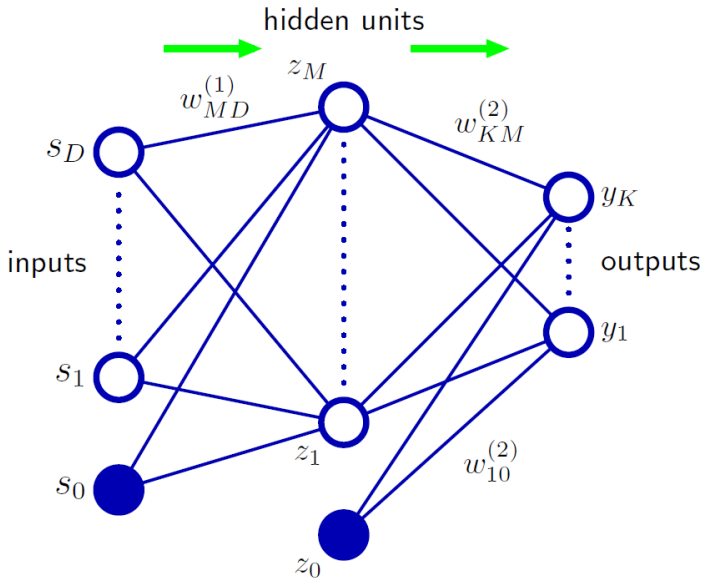



Figure 4.6 A feed-forward neural network. The feed-forward computation is performed from left to right, requiring $D$ input values and resulting in $K$ output values. The type of the output is dependent on the output activation function [121].

to counteract gradient flow problems during Backpropagation Through Time (BPTT), LSTM units are introduced as well.

### 4.3.1 Feed-Forward Neural Networks

As already mentioned in the previous section, SVMs are based on linear combinations of fixed nonlinear basis functions. In contrast to SVMs, NNs consist of a set of basis functions, such that each one is a nonlinear function of a linear combination of its inputs [121]. The coefficients of the linear combination represent adjustable parameters, also known as *weights*, which are adjusted during the training phase of a NN.

Consider the illustration in Figure 4.6, displaying a fully connected NN with three layers. A NN can be described by a set of *neurons*, also known as *units* or *nodes*, that are connected with each other by the use of directed edges. A NN might consist of several layers. The first layer is usually referred to as *input layer*, the last layer as *output layer* and the layer(s) in between as *hidden layer(s)*.

The computation of the output of each neuron is determined in two steps. Let $s_1, \ldots, s_D$ be the input variables to the neuron and $j = 1, \ldots, M$ the number of linear combinations. First, a linear combination of inputs $s_j$ and weights $w_{ji}$ is computed for each neuron, also known as *net activation* or *incoming activation* and denoted as $a_j$. Following, a

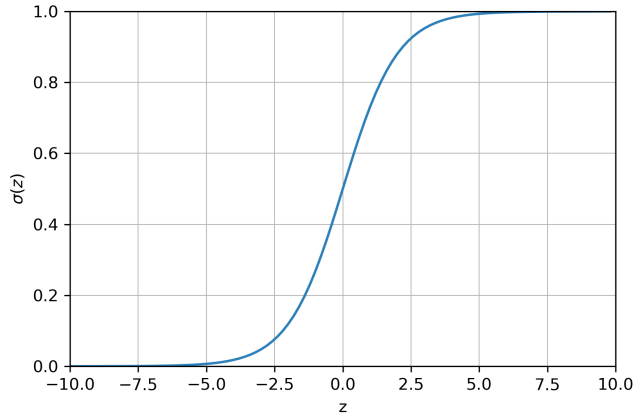



Figure 4.7 Sigmoid activation function.

bias $b_j$, often also denoted as $w_{j0}$, is added and the output $z_j$ is computed using an appropriate activation function $h(\cdot)$ which is dependent on the task. The weight $w_{ji}^{(1)}$ denotes the directed edge to neuron $j$ from neuron $i$. The following equation shows the calculation of the net activation for each single neuron:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} s_i + w_{j0}^{(1)} \tag{4.9}$$

Each activation then gets transformed to an output by a nonlinear activation function $h(\cdot)$:

$$z_j = h(a_j) \tag{4.10}$$

The choice of activation function is highly dependent on the application of the neural network and the underlying data set. As stated in [125], multiclass classification usually uses a *softmax* output activation function, whereas in the case of binary and multilabel classification *sigmoid* is used. The sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-z}}$ used in this work is shown in Figure 4.7.

The feed-forward propagation is successively executed, starting at the input layer up to the last layer. The following equation illustrates the computation of the overall network functions using a *sigmoid* output layer activation function and the architecture of the NN displayed in Figure 4.6:

$$\hat{y}_k(s, w) = \sigma\Big(\sum_{j=1}^{M} w_{kj}^{(2)}\ h\Big(\sum_{i=1}^{D} s_{ji}^{(1)} + w_{j0}^{(1)}\Big) + w_{k0}^{(2)}\Big) \tag{4.11}$$

where $k \in R^K$ and $K$ denotes the number of output neurons.

#### Backpropagation

In order to achieve a well-performing NN, the weights and biases have to be adjusted after each feed-forward propagation. The performance of the NN can be determined by an error function that computes the error between the targets and the predicted output values. Similar to the choice of the activation function, the error function is highly dependent on the output type, such as binary classification, multiclass classification or regression.

Consider a training data set $S = \{s_1, \ldots, s_N\}$ with $N$ observations. The following equation shows a commonly used error function known as *sigmoid cross entropy loss*, adapted to multilabel classification, where $i = 1, \ldots, |d|$ denotes the labels:

$$loss(\hat{y}, y) = \frac{1}{|d|} \sum_{i=1}^{|d|} -(y_i\, log(\hat{y}_i) + (1 - y_i)\, log(1 - \hat{y}_i)) \tag{4.12}$$

Given the error function, the goal is to adjust the weights and biases such that the error is minimized. When choosing an error function that is convex, such as the error function in Equation 4.12, a method known as gradient descent can be applied to minimize the error function and find a local or global *minima*. In the context of NNs, this process of minimizing the error function through gradient descent is known as *backpropagation* or *error backpropagation*. The details are omitted for brevity, but can be found in [121, Chapter 5].

### 4.3.2 Recurrent Neural Networks

Measurement data used for state estimation is collected at several time intervals and is therefore temporal in nature. Recent research has shown that RNNs are well suited for sequential data [125].

A simple RNN consists of an input layer, a hidden layer (i.e., state or memory) and an output layer. At time $t$, input to the network is denoted as $s^{(t)}$, output as $\hat{y}^{(t)}$ and $h^{(t)}$

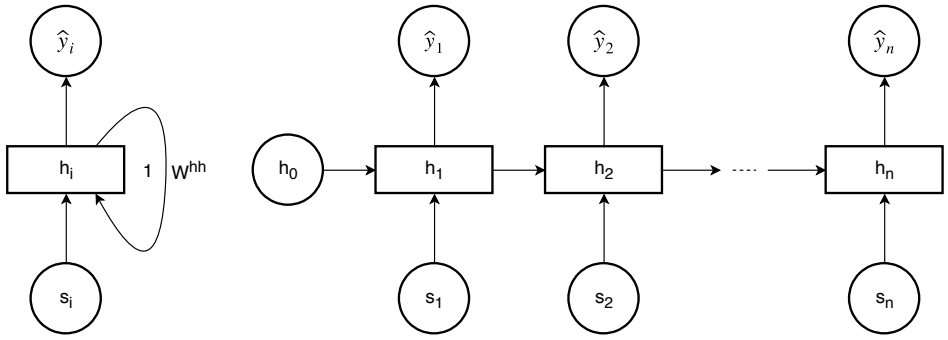



Figure 4.8 The left side shows an RNN in folded form and the right side in unfolded form.

gives the hidden node values. The relationship between input, hidden layer and output is given by the following set of equations:

$$\begin{aligned} h^{(t)} &= \sigma(W^{hs}s^{(t)} + W^{hh}h^{(t-1)} + b_h) \\ \hat{y}(t) &= \sigma(W^{yh}h^{(t)} + b_y) \end{aligned} \tag{4.13}$$

where $W^{hs}$, $W^{hh}$ and $W^{yh}$ denote the weight matrices and $b_h$ and $b_y$ the bias terms.

Figure 4.8 displays an RNN in both folded and unfolded form. As can be seen, the hidden state is forward propagated through the entire network for as many timesteps, as specified. It has to be noted, that the hidden state may consist of not only one hidden node, but several hidden notes and even several hidden layers. As shown later, the hidden nodes can be replaced by specific units, such as Gated Recurrent Unit (GRU) units or LSTM units to improve the performance for learning long-range dependencies.

Similar to the feed-forward NN, the weights and biases of the RNN are adjusted through backpropagation. However, due to the temporal component, the process is slightly different and is referred to as Backpropagation Through Time (BPTT). The details of BPTT are omitted for brevity, but can be found in [126].

Consider a data set consisting of 1000 sequential data points, where each data point comprises 41 features. Thus, the data set is of size $1000 \times 41$. In the case of BPTT, the RNN can be unfolded across all 1000 timesteps and the 1000 input values are forward propagated, resulting in 1000 output values. Subsequently, the error function is used to compute the error and backpropagation to calculate the new gradients that are being backpropagated from the 1000th timestep up to the first one. As can be seen, due to the span of 1000 timesteps and the structure of a simple RNN, the gradients are subject to many modifications and are therefore prone to either vanish or explode. One solution to this problem is truncated BPTT, where the data set is split into several

chunks (e.g., of size 10, resulting in a data set of size $100 \times 10 \times 41$) and BPTT is applied separately to each chunk of data. A different solution is to modify the structure of hidden states, such that gradients can flow easily without being subject to a large number of modifications. One of these concepts can be realized using LSTM units, which are investigated next.

### 4.3.3 Long Short-Term Memory Units

Because simple RNNs are subject to either vanishing or exploding gradients, LSTM cells [127] can be used to overcome this gradient backflow problem and learn long-range temporal dependencies. This is achieved by creating a cell state that makes it easy for information to flow through and not being subject to large modifications.

In one of the first and most prominent works, Hochreiter *et al.* [127] proposed an LSTM unit embodying a Constant Error Carousel (CEC) which ensures a constant error flow. In addition to the unit itself, also known as *memory cell*, two gates, the *input gate* and *output gate*, were introduced.

Gers *et al.* [128] enhanced the aforementioned memory cell by suggesting an additional *forget gate*, which enables the cell to specifically remember or forget information. This allows the cell to reset memory blocks, when content is not needed anymore.

In addition to the exchange of the hidden state $h^{(t)}$, also the cell state $cs^{(t)}$ is exchanged between adjacent LSTM cells. A modern LSTM with forget gates can be described by the following set of equations:

$$\begin{aligned} g(t) &= \phi(W^{gs}s^{(t)} + W^{gh}h^{(t-1)} + b_g) \\ i^{(t)} &= \sigma(W^{is}s^{(t)} + W^{ih}h^{(t-1)} + b_i) \\ f^{(t)} &= \sigma(W^{fs}s^{(t)} + W^{fh}h^{(t-1)} + b_f) \\ o^{(t)} &= \sigma(W^{os}s^{(t)} + W^{oh}h^{(t-1)} + b_o) \\ cs^{(t)} &= g^{(t)} \odot i^{(t)} + cs^{(t-1)} \odot f^{(t)} \\ h^{(t)} &= \phi(cs^{(t)}) \odot o^{(t)} \end{aligned} \tag{4.14}$$

where $W$-terms are the weight matrices, $b$-terms the biases, $g^{(t)}$ is the input run through a *tanh* activation function, $i^{(t)}, f^{(t)}, o^{(t)}$ are the input, forget [128] and output gates, $cs^{(t)}$ is the internal state and $\odot$ indicates an element-wise multiplication.

## 4.4 Preprocessing

This section gives an overview of the considered preprocessing methods in this thesis. The performance of different methods is investigated in Section 6.3.

Consider an input data set $S = \{s_1, \ldots, s_N\}$ comprising $N$ samples with $m$ features. The four investigated preprocessing methods are defined as follows:

### Standardization:

Using standardization, all features $f \in \mathbb{R}^m$ are standardized to have zero mean and unit variance, according to the following equation:

$$f' = \frac{f - \mu(f)}{\sigma(f)} \quad \forall f \in \mathbb{R}^m \tag{4.15}$$

where $\mu(f)$ denotes the mean of the feature vector and $\sigma(f)$ the standard deviation.

### Normalization:

Scaling a vector to unit norm is referred to as normalizing the vector by its Euclidean length. This can either be done per feature or per sample:

**Feature normalization:**

$$f' = \frac{f}{||f||} \quad \forall f \in \mathbb{R}^m \tag{4.16}$$

**Sample normalization:**

$$s' = \frac{s}{||s||} \quad \forall s \in \mathbb{R}^N \tag{4.17}$$

### Min-Max normalization:

Min-max normalization is defined as rescaling each feature vector to the range $[0, 1]$ according to the following:

$$f' = \frac{f - \min(\mathrm{f})}{\max(\mathrm{f}) - \min(\mathrm{f})} \quad \forall f \in \mathbb{R}^m \tag{4.18}$$

Which method to use for which data set highly depends on the particular data set itself. In general, cross-validation can be applied to determine the appropriate preprocessing method. When it comes to preprocessing of training, validation and test data, it must be ensured that first the data sets are split, then preprocessing is applied to the training set and the necessary preprocessing parameters are extracted. Only then the validation and test set are preprocessed with these exact same preprocessing parameters, that have been extracted from the training set.

## 4.5 Imbalanced Classes and Performance Metrics

Especially in the case of intrusion detection, where attacks are usually the exception, the data set might contain a lot more normal data points than abnormal ones. In such a scenario and when modeling the problem as a binary classification problem, the data set is said to have *imbalanced classes*. This class imbalance has to be taken into consideration when defining performance metrics for the classifiers. In this section, performance metrics are defined under consideration of class imbalance. The importance of well chosen performance metrics is illustrated by the example of only using *accuracy* as means to measure the performance and having a data set with two highly imbalanced classes. To get started, basic terminology [129] is defined.

Consider a binary classification problem, where *(P) positive* defines an attack and *(N) negative* no attack. When both the actual and the predicted targets are known, the following metrics can be defined:

- *True positive (TP):* The predicted target is classified as positive and the classification is correct. E.g.: The measurement is classified as an attack and there was an attack.

- *True negative (TN):* The predicted target is classified as negative and the classification is correct. E.g.: The measurement is classified as no attack and there was no attack.
- *False positive (FP) or Type I error:* The predicted target is classified as positive, but the classification is false. E.g.: The measurement is classified as attack, when in reality there was no attack.
- *False negative (FN) or Type II error:* The predicted target is classified as negative, but the classification is false. E.g.: The measurement is classified as no attack, when in reality there was an attack.

The *true positive rate* is defined as the true positives TP divided by the total number of positives $\mathrm{TP} + \mathrm{FN}$ and the *false positive rate* as the false positives FP divided by the total number of negatives $\mathrm{FP} + \mathrm{TN}$.

*Accuracy* represents a metric that defines the proportion of true results out of the total number of results and is defined as follows:

$$\text{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{4.19}$$

A common way to display all of these metrics in a single plot is to use a *confusion matrix*.

Now that basic terminology and accuracy are defined, the problem of only choosing accuracy as metric, can be illustrated. This example is furthermore used in the remaining part of this section to motivate different performance metrics. Consider a data set comprising 50 000 samples and the corresponding binary classification result as illustrated in Table 4.1.

Table 4.1 Data set with highly imbalanced classes and the classification result.

| Metric | Count |
|---|---|
| P | 1000 |
| N | 49 000 |
| TP | 100 |
| TN | 49 000 |
| FP | 0 |
| FN | 900 |

By relating to the previously defined terminology, it can be seen that the data set consists of more non-attacks than attacks. When computing the accuracy according to the previous equation, the result is 98.2%. One might think, by only looking at the accuracy, that this classifier is performing very well. When looking at the attacks, however, it can be seen that only 100 out of 1000 attacks have been correctly detected.

Moreover, a classifier could be constructed that simply classifies every sample as not attacked. Although that classifier is very simple and not sophisticated at all, the resulting accuracy would be $\frac{49\,000}{50\,000} = 98\%$. This emphasizes the need for meaningful performance metrics in the case of highly imbalanced classes.

Two more commonly used performance metrics are *recall* and *precision*, which are defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.20}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.21}$$

In words and relating to the attack detection example, precision would be defined as a metric of how many of the detected attacks actually were attacks, and recall as how many of all attacks were actually detected. In the example above, precision would be 100%, because out of all 100 classified attacks, 100 were actually attacks. Recall, however, would be only 10%, because out of all 1000 attacks, only 100 attacks were actually detected. This definition of precision and recall leads to the so-called $F_1$ score, which is the harmonic mean between the both. The $F_1$ score is applicable for imbalanced classes, when applied correctly, and defined as follows:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.22}$$

The resulting $F_1$ score in the attack detection example would be 0.18%, showing that the classification result is not as good as accuracy represents it.

Another well-suited performance metric for imbalanced classes, used in this work, is the *Area Under Curve (AUC)* of the *Receiver Operating Characteristic (ROC)*. ROC curves feature the true positive rate on the Y axis and the false positive rate on the X axis. The ideal point, a false positive rate of zero and a true positive rate of one, is exactly in the top left corner, which would maximize the area under the curve. The

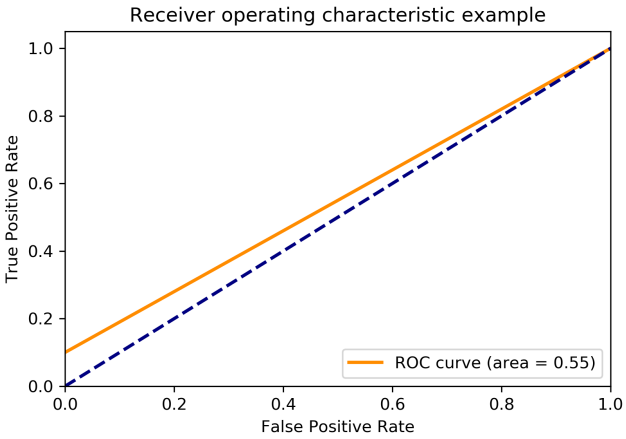



Figure 4.9 Receiver operator characteristic and area under curve for the attack detection example consisting of highly imbalanced classes as shown in Table 4.1.

worst result for the area under curve is 0.5. Figure 4.9 displays ROC and AUC for the attack detection example with the classification result shown in Table 4.1.

In the case of multilabel classification, each performance metric is calculated individually per class. The overall performance metric is then obtained by computing the arithmetic mean of all individual metrics, which is also known as macro-averaging [130].

# Chapter 5

# Test Framework and Data Generation

In previous chapters, the background regarding static state estimation, false data injection attacks and different supervised machine learning algorithms was established. Moreover, a thorough literature research on state of the art works presented already existing methods and approaches.

In this chapter, an overview of the proposed test framework and its individual components is given. Next, the generation of data sets based on real-world load data from the New York Independent System Operator (NYISO) [131], is discussed. Both the construction of measurement data and FDIAs are explained. Because one of the major contributions of this work is the analysis of FDIAs on a subset of state estimation variables, the construction of such attacks and the corresponding labels are investigated. Additionally, various types of attack strength are considered and analyzed. This chapter concludes with an overview of additional data generator features.

## 5.1 Test Framework

One major contribution of this work is the proposed test framework and data generator. The framework itself is divided into several different components according to the concept of object-oriented programming [132]. A high level overview of all components and their interfaces is given in Figure 5.1.

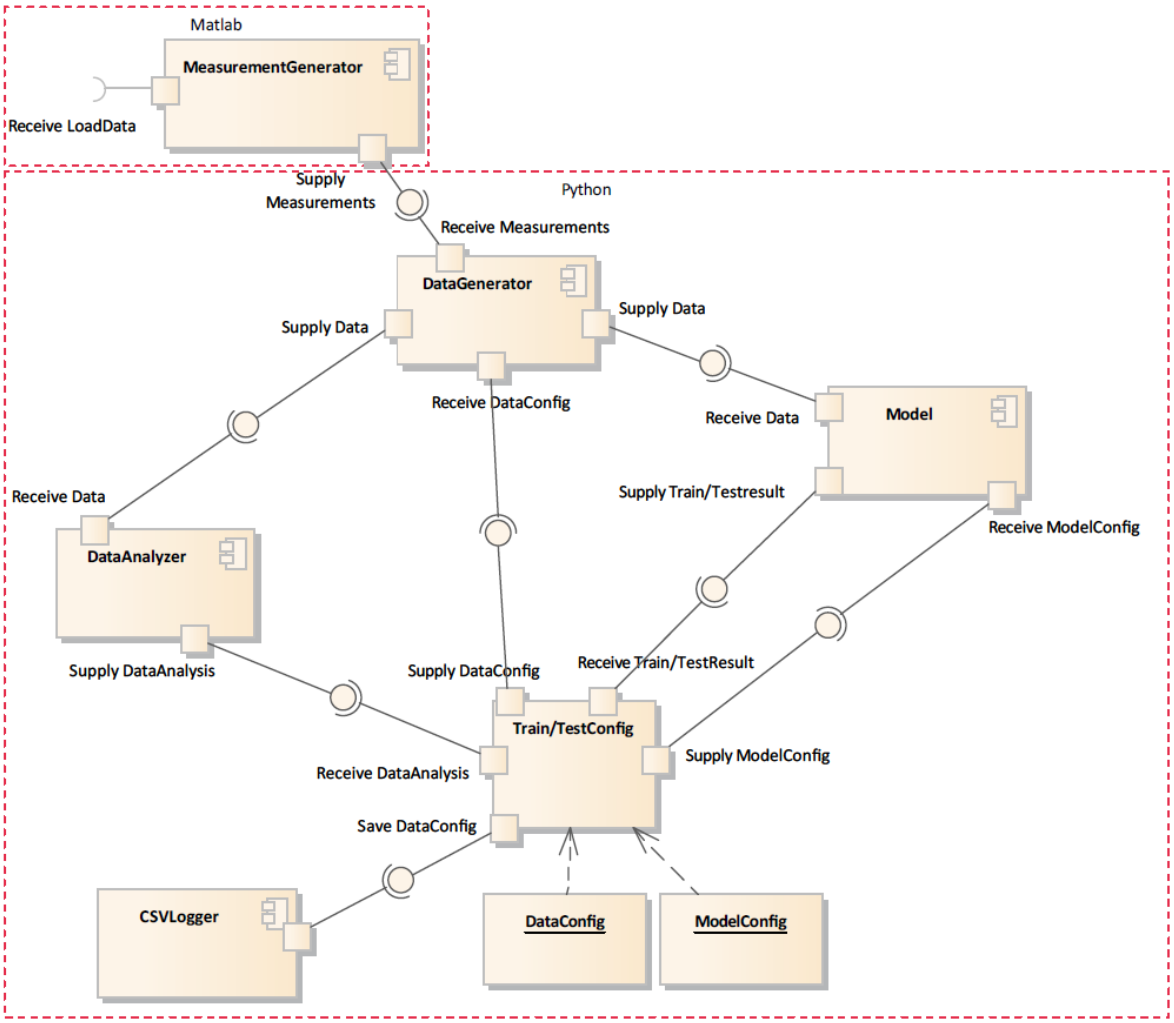



Figure 5.1 Test framework.

Table 5.1 gives an overview of the software used for the implementation of the test framework and the versions in particular.

The individual components of the test framework are described in more detail below.

### MeasurementGenerator

The MeasurementGenerator features two interfaces. As can be seen in Figure 5.1, load data is required in order to generate the measurements. More information is found in Section 5.2.1.

Table 5.1 Used software and the corresponding versions.

| Software | Version |
|---|---|
| Matlab | R2017a |
| MATPOWER | 6.0 |
| Python | 3.6.5 |
| Pip | 10.0.1 (Python 3.6) |
| Keras | 2.1.6 |
| Matplotlib | 2.2.2 |
| Numpy | 1.14.3 |
| Pandas | 0.23.1 |
| Scikit-learn | 0.19.1 |
| Tensorflow-gpu | 1.4.1 |

#### DataGenerator

In order for the DataGenerator to actually generate data, both the measurements generated by the MeasurementGenerator and the configurations provided by the DataConfig are needed. The DataGenerator is used for the construction of FDIAs, described in more detail in Section 5.2.2. Additional features, definable via the DataConfig, are described in Section 5.2.3.

#### DataAnalyzer

The DataAnalyzer receives the data as input and outputs a detailed analysis. Several different metrics are available and can easily be extended.

#### TrainConfig/TestConfig

The TrainConfig/TestConfig is the main construct holding all required information for the execution of automated model training and testing. As mentioned previously, the software is constructed in a modular way, making it easy to add extensions or implement modifications.

The main difference between TrainConfig and TestConfig is, that the TrainConfig contains data, analysis and results for both training as well as validation. The TestConfig only contains data, analysis and result for testing. In addition, the TestConfig contains the ID of a previously trained model, on which the test data should be evaluated.

The following list gives a more detailed explanation of the different objects comprised within the TrainConfig and TestConfig:

- ModelConfig: The ModelConfig includes all relevant information needed by the Model component, in order to construct, train and test the corresponding model. For each ModelConfig there must be a complementing Model implementation.

  Three different ModelConfigs are available: 1) Tensorflow implementation of an LSTM RNN, 2) Keras implementation of an LSTM RNN and 3) Scikit implementation of an SVM. Due to the modularity of the test framework, an additional model, such as a Tensorflow implementation of an SVM, can be easily added.

- TrainResult/TestResult: After training or testing a specific model, the result can be stored within the DataConfig component.

- TrainData/ValidationData/TestData: This configuration includes all the relevant information in order to generate the data, needed by the DataGenerator.

- DataAnalysis: After data generation, the data can be analyzed by the DataAnalyzer and the result stored within this component.

One major advantage of storing all relevant information and configuration in one place is the ease of dumping everything to a CSV file.

In practice, it is advisable to generate a list containing multiple TrainConfigs. Consider the case, where someone wants to execute tests automatically, such as evaluating different types of attack energy as explained in Section 5.2.2. In such a case, a list containing several TrainConfigs can be specified and each item processed successively. The following enumeration shows how such an automation looks like:

1. Select the first TrainConfig in the list.
2. Generate both the training and validation data set based on the DataConfigs specified in the TrainConfig.
3. Analyze both the training and validation data set and store the results in the TrainConfig.
4. Construct and train the model according to the specified ModelConfig and validate the trained model on the generated validation data set.
5. Save the training and validation results in the TrainConfig.
6. Dump all the information and configuration of the TrainConfig to a CSV file.
7. Optionally, create diagrams displaying training performance, or in case of Tensorflow, save the training and validation history to a Tensorboard.

8. Continue with the next TrainConfig in the list.

## 5.2 Data Generation

To be able to train supervised machine learning algorithms, labeled data, also known as ground-truth data, is needed. As discussed in Section 2.3, the system state of a power grid is estimated from a set of measurements collected from various sensors in the transmission network. In this thesis, the proposed algorithms are evaluated on the IEEE 30-bus system. For some examples, the IEEE 118-bus system is considered as well. Details about the topology and system parameters can be found in [26].

### 5.2.1 Construction of Measurements

Before being able to estimate the state from the measurements, the measurements have to be obtained. In simulations, this is usually done by either collecting or generating load data for all load buses in the system, adapting the generation to match the total load and then running an algorithm that solves the power flow problem and determines the steady-state operating point of the electric power system. If the algorithm finds a solution, the entire power grid can be described completely and all power flows can be determined, such as real and reactive power flow in branches, real and reactive power injection at buses and phase angles at buses. The toolbox used in this thesis to solve the power flow problem is MATPOWER 6.0 [39], implemented in Matlab. After solving the power flow problem, the resulting measurements can be used to obtain the state estimate. For better understanding, the following listing shows the steps needed to generate the measurement data:

1. Determine or specify the load for every load bus in the system.
2. Adapt the generation of all generators to match the total load.
3. Run the power flow algorithm to solve for the steady-state operating point.

Next, the data generation is illustrated by referring to the IEEE 30-bus system. The same methodology, however, can be applied to the IEEE 118-bus system as well. By analyzing the IEEE 30-bus system one can see that the total number of buses is divided upon one reference bus, *24 load buses* and five generator buses. This means that a load has to be specified for every load bus. One common way used in literature for load

generation in a network is to sample load from a uniform distribution in the range of $[0.9L_0 - 1.1L_0]$, where $L_0$ is the base load. Some authors [112] argue, that it is more difficult to detect attacks, when the load data is based on a uniform distribution instead of real-world load data, because changes within a day cannot be significantly clustered. In this thesis, however, load data is based on real-world load data from power grid providers, due to the fact that the goal is to stay as close to real-world scenarios as possible.

The data used in this thesis is the load data provided by NYISO. It can be freely obtained on their homepage [131] and contains the load of 11 load buses in the transmission network, collected at five minute intervals since May 2001. Because in this data set only 11 buses are available and there exist 24 load buses in the IEEE 30 bus system, load data must be replicated, whereby it must be ensured that every bus in the IEEE 30 bus system matches the same bus in the data set for all samples. This guarantees that the distribution of each load bus does not get mixed up between several buses in the data set.

An excerpt of the data file, showing the names of the 11 load buses and the corresponding load (specified in Megawatt (MW)), can be seen in Listing 5.1.

Listing 5.1 Excerpt of the NYISO load data.

```
1  "Time Stamp","Time Zone","Name","PTID","Load"
2  "01/02/2013 00:00:00","EST","CAPITL",61757,1227.7
3  "01/02/2013 00:00:00","EST","CENTRL",61754,1745.2
4  "01/02/2013 00:00:00","EST","DUNWOD",61760,586.8
5  "01/02/2013 00:00:00","EST","GENESE",61753,1024.8
6  "01/02/2013 00:00:00","EST","HUD VL",61758,992.4
7  "01/02/2013 00:00:00","EST","LONGIL",61762,2170.1
8  "01/02/2013 00:00:00","EST","MHK VL",61756,940.2
9  "01/02/2013 00:00:00","EST","MILLWD",61759,343.5
10 "01/02/2013 00:00:00","EST","N.Y.C.",61761,5100.8
11 "01/02/2013 00:00:00","EST","NORTH",61755,823.8
12 "01/02/2013 00:00:00","EST","WEST",61752,1685.4
```

According to the IEEE 30-bus system, the bus CAPITL is assigned to bus 1, 12 and 23. Because 11 buses cannot be replicated exactly upon 24 buses, the remaining $(11 \cdot 3) - 24 = 9$ samples are discarded. Subsequently, due to this replication, the time interval between samples is not five minutes anymore, but 15 minutes. For this thesis, data of the years 2013-2017 (5 years) is chosen. After replication for the IEEE 30-bus system, this results in a data set containing $172\,996 \times 24$ samples with a time interval

of 15 minutes. For the IEEE 118-bus system, which consists of 64 load buses, the data set contains $86\,273 \times 64$ samples with a time interval of 30 minutes.

Now that the load data is replicated, the total generation must be adapted to match the total load by distributing the increase or decrease in generation upon all available generator buses. Moreover, before running the power flow algorithm, the load at every load bus must be scaled by the *baseMVA* of the power system, which is determined by taking the max value of all load buses. If $lb \in \mathbb{R}^k$ represents the load at $k$ load buses, each load is scaled according to the following equation:

$$lb_i = \frac{lb_i}{\text{baseMVA}} \quad \forall i \in \{1, \ldots, k\} \tag{5.1}$$

Again, this is conducted for every sample. Next, the power flow algorithm can be run. Because in this thesis the focus is on DC state estimation, the DC power flow algorithm is run, which considers only real power flows and injections as opposed to real and reactive. For this task, the method runcpf of the MATPOWER toolbox is used which returns load parameters of the entire network including the real phase angles at the buses, which can later be used to validate the resulting phase angles of the static state estimation. The interested reader is referred to the MATPOWER user's manual [133] for a detailed explanation of the results of the DC power flow algorithm.

Now that the real phase angles of the buses and the real power flows in the branches are known, state estimation can be conducted. For clarity, the specific steps are summarized in the listing below:

1. Choose a number of measurements, such that $m > n$.
2. Construct the measurement matrix $H$, which is constant across all samples per power system.
3. Define matrix of meter errors $W$.
4. Estimate the system state according to Equation 2.12.
5. Validate the results by calculating the difference between the real phase angles from the power flow algorithm and the estimated phase angles from the static state estimation.

After running the DC power flow algorithm, the result contains the real power flow in branches from, for example, both bus 1 to bus 2 as well as bus 2 to bus 1. However, because branch resistances and reactive measurements are neglected, both power flows

are the same. Moreover, after running the DC power flow algorithm the measurements have to be scaled again according to Equation 5.1 and according to the $baseMVA$, which is part of the result. In order to construct matrix $H$, only the real power flow in one direction of the 41 branches in the IEEE 30-bus system is considered. The measurement matrix $H$ is then constructed according to the example given in Section 2.3.2. Because the state variables consist of all phase angles except the reference bus, i.e., 29 state variables, the measurement matrix is of size $41 \times 29$. The meter errors are assumed to be 0.0001 for every measurement. Given this information, the state estimate can be computed for each sample.

For practical uses, an algorithm get_state_vars_with_load is implemented in Matlab, which receives as input the CSV files containing the load data from NYISO, a scalar representing the meter errors which are the same for all measurements and a string specifying the IEEE bus system. The algorithm is able to handle arbitrary IEEE bus systems, performs bus replication, solves the power flow problem, computes the state estimate and returns the following results:

Per data set:

- Measurement matrix $H$
- Matrix of meter errors $W$

Per sample:

- Scaled measurements used for static state estimation (i.e., real power flow in one direction of all branches).
- Real phase angles obtained by the power flow algorithm.
- State estimation variables consisting of the phase angles at all buses except the reference bus.
- Difference between real phase angles and estimated phase angles.

Table 5.2 shows the output for four phase angles of one sample in radians where it can be seen that the difference between the real and the estimated phase angle is very small. This indicates that the state estimation is working as expected. Moreover, by examining the measurement matrix $H$, it can be seen that the matrix is extremely sparse with only 80 non-zero elements out of 1189. The largest phase angle out of all samples is 0.3435 rad or 19.68°.

Table 5.2 Real phase angles, estimated phase angles and the difference between both for four different buses of one sample.

| Bus | Real phase angle / rad | Estimated phase angle / rad | Difference / rad |
|---|---|---|---|
| 2 | 0.0145 | 0.0145 | $-4.8572 \times 10^{-17}$ |
| 10 | 0.0796 | 0.0796 | $-4.1633 \times 10^{-17}$ |
| 20 | 0.0343 | 0.0343 | $6.2450 \times 10^{-16}$ |
| 30 | -0.1452 | -0.1452 | $-2.7756 \times 10^{-17}$ |

### 5.2.2 Construction of FDIAs

One of the major contributions of this thesis is the consideration of FDIAs targeting a subset of state estimation variables. In this section, both the construction of subset level FDIAs as well as the creation of corresponding class labels is explained. The creation of FDIAs targeting all state variables are not explained explicitly, because such attacks can easily be obtained by setting the number of subsets to one.

FDIAs can be constructed using the DataGenerator component of the test framework described in Section 5.1.

#### Construction of Subset Level Attacks

As discussed in Section 2.4, $c$ controls the error that gets introduced to the state estimation variables when constructing the attack vector in the form $a = Hc$ and adding it to the original measurements $z$, such that $z_a = z + a$. In case the attacker wants to introduce an error to specific state variables, only the corresponding fields of $c$ must be populated.

Let $d$ denote the number of chosen subsets that all state variables $x \in \mathbb{R}^n$ should be divided upon and $d_i \in \{1, \ldots, d\}$ the subset at the $i$th position. Additionally, $atk \in \mathbb{R}^d$ serves as a vector of indices describing which subsets should be attacked and which not, with 1 representing an attack and 0 representing no attack.

Algorithm 5.2 generically describes the construction of state variable subsets and attacks, given the total number of state variables $n$, the number of subsets $d$, the vector of attack indices $atk$ and the attack value $C$. The algorithm first checks if the set of state variables can be divided exactly upon $d$ subsets or not and determines the appropriate subset sizes. After subset size determination, the vector of attack indices $atk$ indicates whether the subset should be attacked or not. In the case of an attack, the attack value

```
1: procedure CONSTRUCT_SUBSETS(d, n, atk, C)
2:     b ← ⌊n/d⌋
3:     c_{1:n} ← 0
4:     r ← n mod b
5:     if r = 0 then
6:         for i ← 1, d do
7:             if atk_i = 1 then
8:                 c_{(i*b)−(b+1):i*b} ← C
9:             end if
10:        end for
11:    else
12:        for i = 1, d do
13:            if atk_i = 1 and i = d then
14:                c_{(i*b)−(b+1):(i*b)+r} ← C
15:            else if atk_i = 1 then
16:                c_{(i*b)−(b+1):i*b} ← C
17:            end if
18:        end for
19:    end if
20:    return c
21: end procedure
```

Figure 5.2 Algorithm for the construction of $c$ with given number of subsets $d$, state variables $d$, attack index $atk$ and attack value $C$.

$C$ gets introduced to the subset of state variables. After successful construction of $c$, the attack vector can be generated according to Section 2.4.

It has to be noted that $C$ and $c$ are not identical. $c$ always represents a vector of the same size as the number of buses, whereas $C$ can represent either a scalar or a vector. In the case of a scalar, the value of $C$ gets introduced to the vector $c$ by populating the fields that should be attacked, resulting in the same value for each populated field. In the case of a vector, $C$ must be the same size as $c$. It must furthermore be ensured, that in the case of a scalar, $C$ is not zero and in the case of a vector there exists at least one non-zero element in each attacked subset of $c$. Section 5.2.2 discusses the construction of $C$ with regard to the energy of the attack in more detail.

### Definition of Class Labels

From a machine learning point of view, the aforementioned detection problem of FDIAs can be modeled as a binary classification problem. To be able to train supervised

machine learning algorithms, the data has to be labeled. Let the data set $D$ consist of a set of samples $S = \{s_1, \ldots, s_N\}$ comprising $N$ observations of measurement vectors $z$ with $m$ features and a set of labels $Y = \{y_1, \ldots, y_N\}$. It has to be noted, that in case of a time series data, every sample $s_i \in S$ corresponds to a particular timestep $t$, such that $s_i = s^{(t)} \, \forall \, i, t \in \{1, \ldots, N\}$.

In the simpler case of classifying the data as either attacked or not attacked, the corresponding class labels would be defined as follows:

$$y_i = \begin{cases} 0, & \text{if } a_i = 0 \\ 1, & \text{if } a_i \neq 0 \end{cases} \tag{5.2}$$

In the case of subset level detection, the class label is directly correlated to the vector of attack indices $atk$:

$$y = atk \tag{5.3}$$

Thus, $y_i \in \mathbb{R}^d \, \forall \, i \in \{1, \ldots, N\}$. Each subset is considered to be independent of all other subsets. Because of the fact that more than one subset can be attacked at a given time, the output has to be considered as multilabel classification instead of multiclass classification. With multilabel classification, more than one class can be positive for a given output, whereas multiclass implies that at most one class can be positive.

Listing 5.2 gives an example of the labels of five samples where the number of subsets is 10 and each row is representing one sample and each column one subset. As can be seen, samples two and three are not attacked at all, whereas in the other samples five out of 10 subsets are attacked. These parameters can be explicitly specified when constructing the data set as described in Section 5.2.3.

Listing 5.2 Class labels of five samples with ten subsets.

```
1 [1, 1, 1, 0, 0, 0, 1, 0, 0, 1]
2 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
3 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4 [1, 1, 0, 1, 0, 0, 1, 0, 0, 1]
5 [1, 0, 1, 1, 1, 0, 0, 1, 0, 0]
```

#### Energy of FDIAs

The first factor introduced in order to vary the attack strength is $P$, which denotes the fraction of attacked samples $s_i \in \{s_1, \ldots, s_N\}$ out of all samples $N$.

Second, the parameter $C$, as depicted in Algorithm 5.2, determines the error that gets introduced to the state estimation variables. As previously mentioned, $C$ can either be a scalar or a vector. In the simple case of a scalar, $C$ has to be large enough to cause an impact on state estimation, such as $C = 0.2$, but must not be unnecessarily big, such as $C = 20000$. When $C$ is unnecessarily big, the FDIA will still bypass the traditional BDD but will more easily trigger more advanced detection methods and will be detected nonetheless when resulting in phase angles smaller than $-360°$ or larger than $360°$.

In a more advanced case and as typically used in several FDIA studies [110, 117], $C$ can be defined as a vector that is constructed by multiplying a scalar $A$ with a Gaussian based on the mean and variance extracted from the measurement vector $z$. Thus, for a given vector of measurements $z \in R^m$ with a given scalar $A$, the vector $C \in \mathbb{R}^n$ can be constructed as follows:

$$C_i = A\mathcal{N}(\mu(z), \sigma^2(z)) \quad \forall i \in \{1, \ldots, n\} \tag{5.4}$$

By use of this notation, $A$ can be used to scale the strength of the attack.

### 5.2.3 Additional Concepts

This section introduces a method for the construction of three-dimensional data sets out of two-dimensional data sets. Moreover, features and settings of the DataGenerator are introduced.

#### Extracting Batches Using a Sliding Window

In contrast to an SVM that expects input data in the form of (samples × features), from now on referred to as *2D data set*, the RNN introduces the temporal component, thus expecting input data in the form of (samples × timesteps × features), from now on referred to as *3D data set*. This shape of input data is achieved by extracting batches of size $k$ (i.e., timesteps) from the two-dimensional data set using a sliding window of size

$k$ that is successively moved over the data set. The label then represents the existing or not existing attacks in the last time step of the sample consisting of $k$ time steps. Figure 5.3 illustrates this concept. Considering a 2D data set with $N$ samples, after applying the sliding window method the constructed 3D data set consists of $N-(k-1)$ samples.

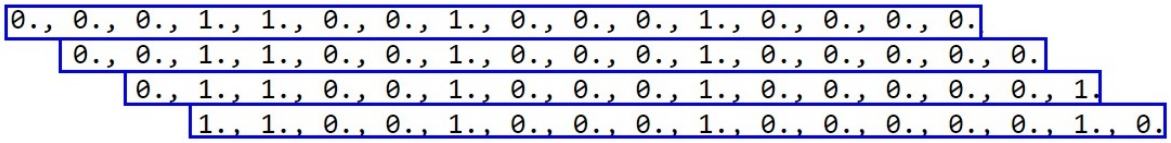



Figure 5.3 Illustration of the sliding window concept with a window of size 16 using output labels consisting of one subset.

### DataGenerator Features

The *DataGenerator* features all previously explained concepts as well as additional ones. An overview of the available features is given in the listing below:

- Specify different numbers of subsets ranging from one up to the total number of state estimation variables.
- Specify how many subsets out of all subsets should be attacked and with what probability.
- Specify the attack energy with a specific value in case of a scalar or $A$ in case of constructing FDIAs based on the distribution of the original measurements. When choosing the construction based on the distribution of the original measurements, the mean can either be specified according to the original measurements or set to a different value.
- Specify the number of timesteps in case of the construction of a *3D data set*.
- $P$
- Alternatively to $P$, one can manually specify which timesteps out of all timesteps should be attacked and how many of them. For example, when specifying 16 timesteps, it can be defined that timesteps 5 and 8 get attacked every time, or that they get attacked by a specific probability. It can furthermore be defined, that out of all 16 timesteps, each time 4 timesteps get randomly chosen and attacked.
- Specify different preprocessing method.

- Construct a 2D or a 3D data set.

# Chapter 6

# Experimental Results

In the previous chapters, the theoretical background, which is needed for the understanding of the experiments in this chapter, was established. This chapter starts by outlining the experimental setup and the proposed models in specific. Next, FDIAs with different types of attack strength and energy are investigated in detail and conclusions are drawn. Multiple preprocessing methods are cross-validated on the proposed models and the best preprocessing method for each model is determined. Additionally, the correlation between preprocessing methods and attack energies is investigated. This chapter concludes by evaluating the performance of the proposed models on FDIAs targeting different numbers of subsets.

## 6.1 Experimental Setup and Proposed Models

In this section, the proposed models and data sets are described. If not stated otherwise, the tests are conducted based on the models described in this section.

All experiments are evaluated on the IEEE 30-bus test system. For the RNN, a 3D data set of size $172\,962 \times 16 \times 41$ is used. The number of 16 timesteps is chosen via cross-validation and represents a time frame of four hours. For the SVM, a 2D data set of size $10\,000 \times 41$ is used. Both data sets are split into 70% training and 30% validation data. In the case of test data, a new 3D data set of size $172\,962 \times 16 \times 41$ is constructed for the RNN and a 2D data set of size $10\,000 \times 41$ for the SVM.

In the case of data sets, where the number of subsets is larger than one, the number of attacked subsets out of all subsets is $\left\lceil \frac{d}{2} \right\rceil$. The performance metrics stated throughout

this chapter are always determined by evaluating the proposed model on the validation data set and not the training data set.

**SVM**

Scikit-learn in Python is used to implement the SVM with a radial basis function kernel. The hyperparameters $C$ and $\lambda$ are determined using a grid search method in a user-defined interval. The best performing parameters are $C = 10\,000$ and $\lambda = 0.1$.

**RNN**

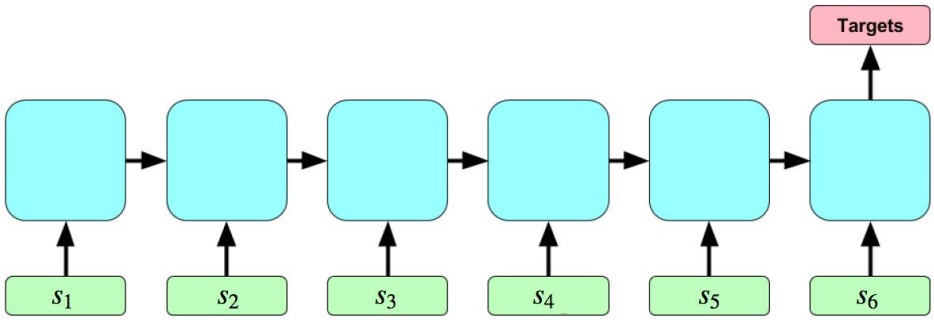



Figure 6.1 Recurrent neural network with a many-to-one architecture.

The used RNN reflects a many-to-one architecture, as illustrated in Figure 6.1. As can be seen, the label, and therefore the error, is only applied to the last time step out of the $k$ time steps (16 in this case). As error function, *sigmoid cross entropy loss* adapted to multilabel classification is used, as shown in Equation 4.12.

The RNN consists of one dense input layer of size $m$, two stacked LSTM layers of size 200 and one dense output layer of size $d$. Dropout of 30% is applied to the non-recurrent connections in order to avoid overfitting. The RNN is implemented in Python using TensorFlow. Different methods for stochastic gradient-based optimization have been evaluated and Adam [134] is chosen. Moreover, the learning rate starts at a predefined value and decreases over time by a given decay value of 0.93. Training is stopped, when the training cost is below 0.001 for two epochs or when the validation cost did not increase by 0.001 for two epochs. The maximum number of epochs, when training is stopped nonetheless, is 200. The chosen batch size is 256.

All hyperparameters, such as number of hidden layers, size of hidden layers, batch size and more are chosen using cross-validation. The details, however, are omitted for brevity.

The SVM is trained on a notebook with an Intel i5-5300U@2.3 GHz CPU and 8 GB RAM and the RNN on an NVIDIA PNY Tesla K40c GPU with 12GB GDDR5 RAM.

## 6.2 Analysis of Different FDIAs

In this section, a simple analysis of FDIAs constructed with different attack energies and numbers of subsets is conducted. Due to the fact that the data sets consist of more than three dimensions, Principal Component Analysis (PCA) is performed for dimensionality reduction and the first two principal components are plotted showing both normal and attacked measurements. Moreover, it is evaluated how many principal components are needed to obtain 90% of the explained variance.

In Section 6.2.1, only attacks that introduce an error to all state variables are considered and evaluated with different attack energies. In Section 6.2.2, the impact of varying the number of subsets is investigated.

### 6.2.1 Analysis of FDIAs Targeting all State Variables

As mentioned in Section 5.2.2, $C$ can either be a scalar or a vector. Four variations of attack energy are considered, covering both cases of introducing a smaller (i.e., $\text{mean}(C) = 0.0024$) and larger (i.e., $\text{mean}(C) = 0.24$) error to the state estimation variables. Both $A_1$ and $A_3$ as well as $A_2$ and $A_4$ have been determined such that they are similar in terms of attack energy. The four variations are listed below:

$A_1$: $C = 0.24$

$A_2$: $C = 0.0024$

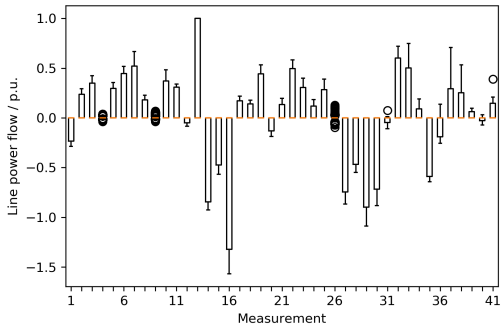
$A_3$: $C_i = 10\,\mathcal{N}(\mu(z), \sigma^2(z)) \quad \forall i \in \{1, \ldots, n\}$

$A_4$: $C_i = 0.1\,\mathcal{N}(\mu(z), \sigma^2(z)) \quad \forall i \in \{1, \ldots, n\}$

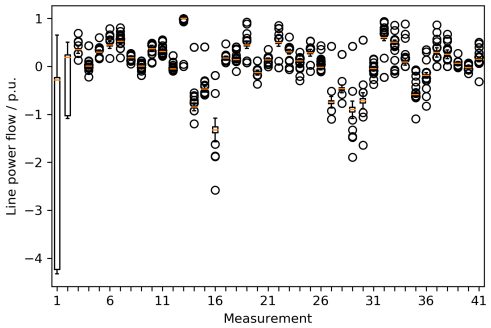
In order to evaluate the impact of FDIAs on original measurements, the data sets are not preprocessed. An evaluation of different preprocessing methods can be found in Section 6.3.

All data sets are constructed based on the IEEE 30-bus test case. $P$ is set to 0.5, resulting in a data set that has as many attacked as non-attacked measurements.

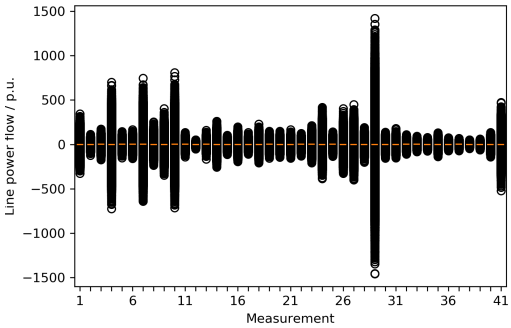
As a first investigation, the impact of attacks is examined by comparing boxplots for each measurement for both original as well as attacked measurements, as shown in Figure 6.2. Figure 6.2a shows the original measurements, Figure 6.2b measurements




(a) Original, untampered measurements.




(b) $A_1$, i.e., $c = 0.24$.




(c) $A_3$, i.e., $A = 10$.

Figure 6.2 Boxplot of original and attacked measurements, illustrating the impact of FDIAs with different attack energies on the original measurements.

after adding FDIAs using the attack energy in $A_1$ and Figure 6.2c measurements after adding FDIAs using the attack energy in $A_3$.

It can be seen that the original measurements span a range of -1.5 to 1.0. Constructing FDIAs with the attack energy scheme in $A_1$ largely influences the first two measurements, resulting in a clear difference from the original measurements. As seen earlier, the attack vector is based on a linear combination of the measurement matrix $H$ and the introduced error $c$, i.e., $z_a = z + Hc$. In the case of $A_1$, the range of the attacked measurements is very close to the range of the original measurements. In the case of $A_3$, however, the attacked measurements become large, with values greater than 1000 and smaller than $-1000$. This is due to the fact, that $C$ in $A_3$ is based on the assumed normal distribution of the original measurements, which has a mean of 0.244 and a standard deviation of 4.62 in the case of this data set. Combined with the large values in the measurement matrix $H$, such as values in the range of 50, the multiplication factor $A = 10$ and the large standard deviation in the original measurements, attacked

measurements in $A_3$ are much larger than the original measurements. On the other side, although, attacked measurements in $A_3$ are based on the same distribution as the original measurements, which makes them harder to be detected when data sets are transformed to a similar range of values. This behaviour is investigated when evaluating the proposed detection methods in Section 6.4.

As a second means of evaluation, descriptive statistical values, the histogram of the first two principal components and a scatter plot showing the first two principal components is given for each data set.

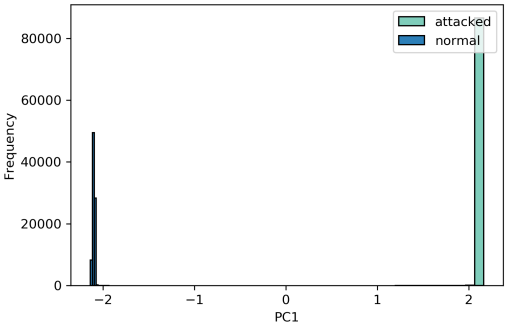
Table 6.1 Mean value of $C$, max and min value of $z$, explained variance of the first two principal components and the minimum required number of principal components such that the sum is $\geq 90\%$ for FDIAs targeting all state variables with four different attack energy schemes.

| FDIA | mean($C$) / rad | max($z$) / p.u. | min($z$) / p.u. | PC1 | PC2 | #PCs$\mid \sum \geq 90\%$ |
|---|---|---|---|---|---|---|
| $A_1$ | 0.24 | 1 | -4.34 | 99.33% | 0.58% | 1 |
| $A_2$ | 0.0024 | 1 | -2.58 | 84.3% | 9.4% | 2 |
| $A_3$ | 0.24 | 1367.25 | -1519.81 | 40.6% | 19.16% | 9 |
| $A_4$ | 0.0024 | 13.54 | -15.46 | 40.3% | 19.19% | 9 |

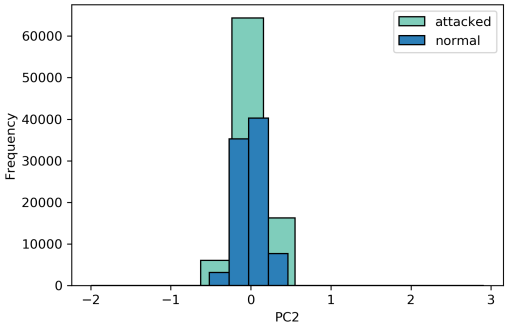
Table 6.1 gives an overview of the mean value of $C$, the maximum and minimum value of $z$, the explained variance of the first two principal components and the minimum required number of principal components such that the sum is $\geq 90\%$. As can be seen, the mean value of $C$ in the cases of $A_1$ and $A_2$ corresponds exactly to the specified scalar. In the cases of $A_3$ and $A_4$ the mean value of $C$ depends on the distribution of the actual measurements. It has to be noted, that $C$ represents the actual error introduced to the state estimation variables prior to the normalization of the data sets. It can furthermore be seen, that the smaller the introduced error is, the more principal components are needed to account for 90% of the variance. Likewise, more principal components are needed when $c$ is constructed based on the distribution of the measurements, in contrast to a fixed scalar. As already seen previously, the values of the measurements in $A_3$ and $A_4$ span a much larger range than in $A_1$ and $A_2$.

Figures 6.3 and 6.4 show histograms displaying the distribution of the first two principal components for all four attack scenarios. Although not all principal components represent a meaningful amount of explained variance, e.g., Figure 6.3b only represents 0.58%, all histograms are displayed for the sake of completeness. It can be seen, that in the case of $A_1$ and $A_2$, the attacked measurements comprise a more narrow range of values than in the case of $A_3$ and $A_4$, where the attacked measurements are based

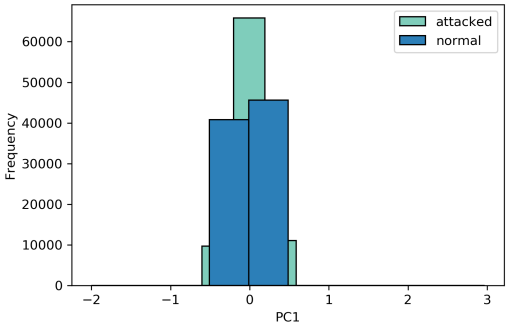
on the distribution of the original measurements. It is furthermore shown, that when the error is large, such as in $A_1$, it is easier to distinguish attacked measurements from normal measurements than when the error is small. It has to be noted, however, that in the case of $A_3$ and $A_4$, the first two principal components only comprise about 60% of explained variance. This leads to the suggestion, that when building machine learning models for the detection of attacks, not only the first two principal components should be used as features. Another meaningful investigation would be the impact of different preprocessing methods on the distribution of the first two principal components. This research, however, is omitted for brevity, but an investigation of different preprocessing methods on the performance of the proposed methods can be found in Section 6.3.



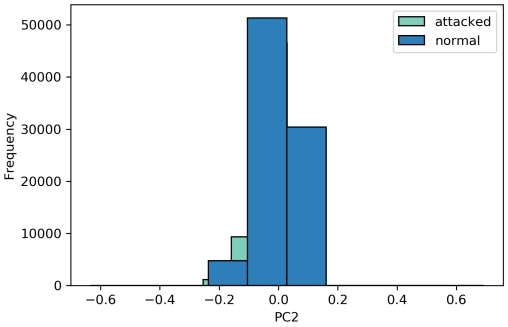

(a) $A_1$, i.e., $C = 0.24$, PC1.




(b) $A_1$, i.e., $C = 0.24$, PC2.




(c) $A_2$, i.e., $C = 0.0024$, PC1.

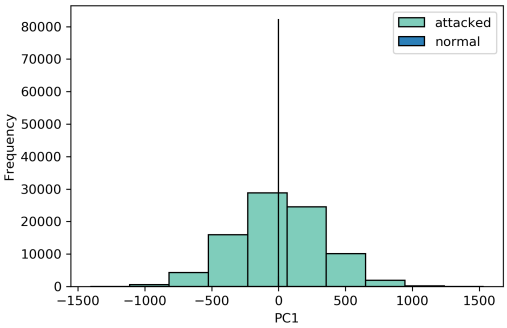



(d) $A_2$, i.e., $C = 0.0024$, PC2.

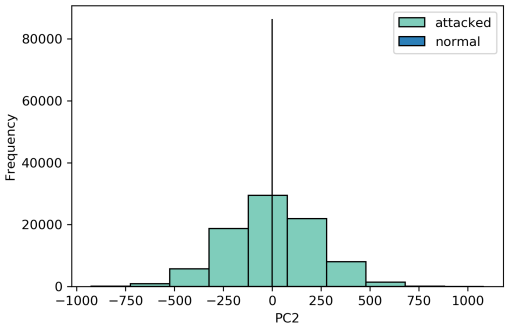
Figure 6.3 Distribution of attacked and normal measurements with $C$ based on a fixed scalar.

In addition to the histograms, scatter plots showing the first two principal components, displaying attacked and normal measurements, can be seen in Figure 6.5.

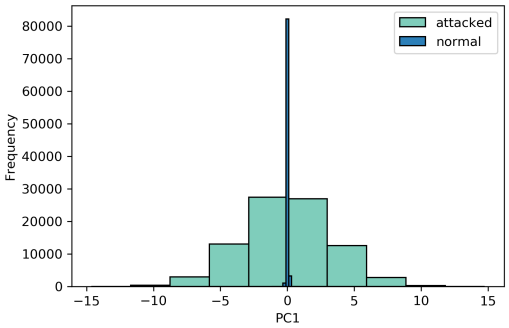
As already illustrated in Figure 6.3a, it can also be seen in Figure 6.5a that attacked measurements and non-attacked measurements can be clearly separated. Figure 6.5b shows that when the error gets smaller, normal and attacked measurements are highly



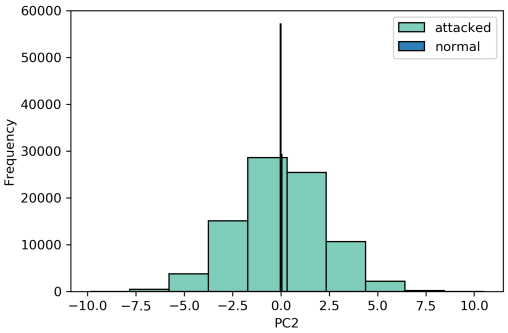

(a) $A_3$, i.e., $A = 10$, PC1.




(b) $A_3$, i.e., $A = 10$, PC2.




(c) $A_4$, i.e., $A = 0.1$, PC1.



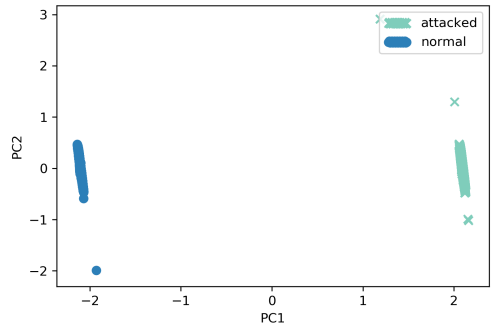

(d) $A_4$, i.e., $A = 0.1$, PC2.

Figure 6.4 Distribution of attacked and normal measurements with $C$ based on the distribution of the measurements.

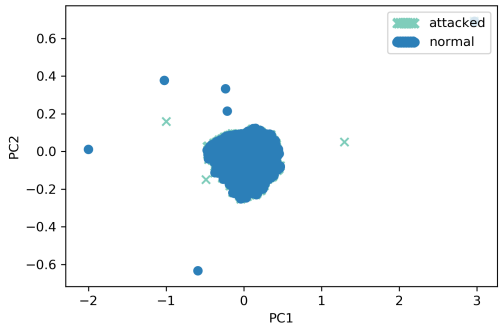
overlapping. One general conclusion that can be drawn is, that, when either the introduced error gets very small or FDIAs are constructed based on the distribution of original measurements, either the number of features must be increased or the complexity of the proposed classification models. It can be seen well, that in the case of Figure 6.5b, 6.5c and 6.5d and when using only two principal components, a simple clustering algorithm such as kNN does not suffice to distinguish between attacked and non-attacked measurements. In that case, either more features or a more complex model is needed.

### 6.2.2 Analysis of FDIAs Targeting Subsets of State Variables

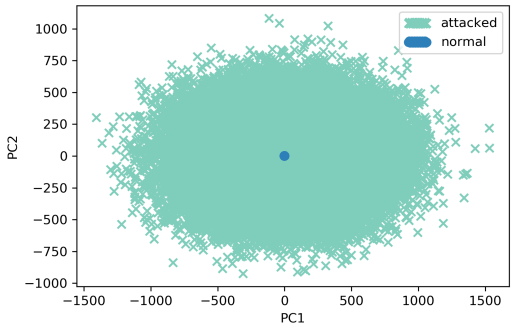
A similar comparison in terms of comparing histograms of the first two principal components as in the previous section is difficult when the number of subsets increases, because the number of histograms and scatter plots increases as well and a comparison



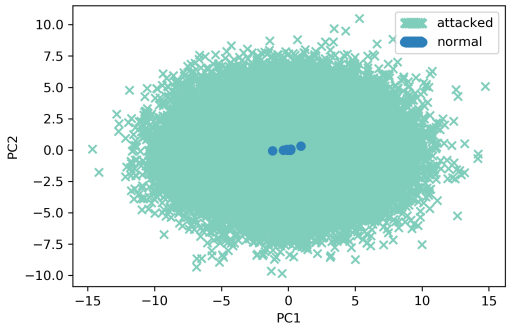

(a) $A_1$, i.e., $C = 0.24$.




(b) $A_2$, i.e., $C = 0.0024$.




(c) $A_3$, i.e., $A = 10$.




(d) $A_4$, i.e., $A = 0.1$.

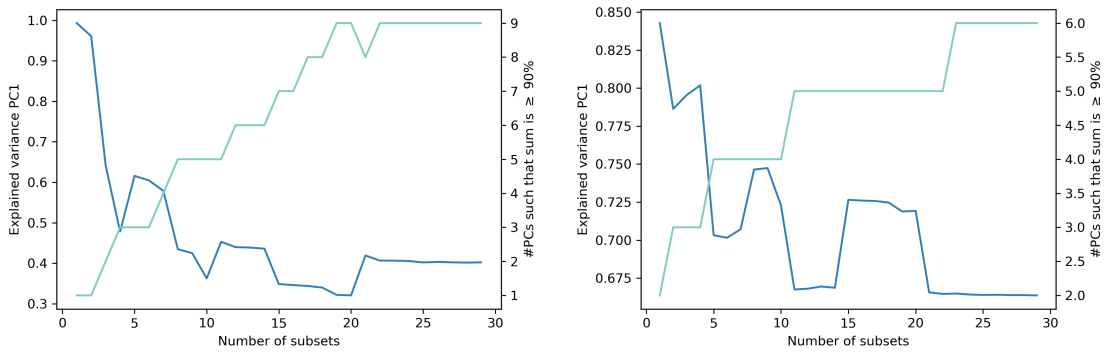
Figure 6.5 Scatter plots showing the first two principal components of both attacked and normal measurements with different attack energies.

becomes confusing. Thus, in this section, different numbers of subsets are investigated according to the following defined metrics:

- Explained variance of the first principal component.
- Number of principal components needed, such that the sum is $\geq 90\%$.

Figure 6.6 displays the aforementioned metrics for the attack energy schemes $A_1$ and $A_2$. As can be seen, the amount of explained variance of the first principle component decreases when the number of subsets increases. Although the decrease fluctuates, the trend can be clearly seen. At the same time, the minimum required number of principal components such that the sum is $\geq 90\%$ increases, which is to be expected. Interestingly, the amount of explained variance for the first principal component when investigating the attack energy schemes $A_3$ and $A_4$ stays around 40% and the minimum

required number of principal components such that the sum is $\geq 90\%$ is 9, regardless of the number of subsets. Thus, the plots have been omitted.




(a) $A_1$, i.e., $C = 0.24$.

(b) $A_2$, i.e., $C = 0.0024$.

Figure 6.6 Explained variance of the first principal component (decreasing line) and the minimum required number of principal components such that the sum is $\geq 90\%$ (increasing line) for different numbers of subsets.

This concludes the analysis of different FDIAs. In the next sections, the performance of the proposed models and different preprocessing methods will be investigated.

## 6.3 Evaluation of Different Preprocessing Methods

In this section, both the SVM and RNN are evaluated on different preprocessing methods with data sets containing FDIAs with different attack energies.
This section contains an extensive evaluation, leading to conclusions not only concerning preprocessing methods, but also confirming the conclusions that were drawn in Section 6.2, when evaluating different data sets.

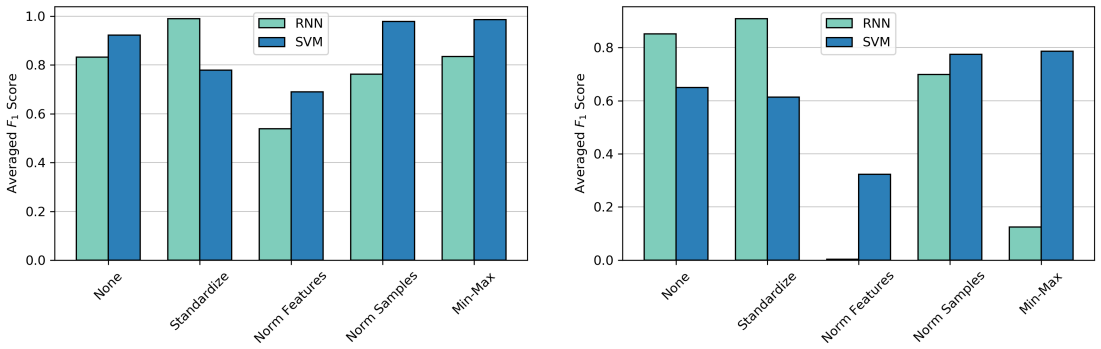
In order to cover all ranges of attack energies, preprocessing methods and numbers of subsets, the following variations of attacks are investigated:

- **Preprocessing:** None, standardization, feature normalization, sample normalization and min-max normalization.
- **Attack energies:** All four attack energies $A_1$, $A_2$, $A_3$ and $A_4$ as defined in Section 6.2.1.
- **Attack strengths** $P$**:** 0.05, 0.1, 0.3 and 0.5
- **Numbers of subsets** $d$**:** 1, 10, 20, 29

This results in $5 \cdot 4 \cdot 4 \cdot 4 = 320$ variations for each SVM and RNN, resulting in 640 variations in total. Due to the available test framework, all tests can be defined and executed automatically.

As a first evaluation, all preprocessing methods are evaluated for both SVM and RNN separately for both types of attack energies based on either the scalar or the distribution of the original measurements. This results in the evaluation of 20 different variations, whereby each variation considers all four numbers of subsets and all four attack strengths. Figure 6.7 shows the results of this evaluation in terms of averaged $F_1$ score of all tests, grouped by the type of attack energy for both SVM and RNN.

It can be seen, that using standardization the RNN outperforms the SVM for both types of attack energy, whereas the SVM outperforms the RNN when using sample normalization and min-max normalization. The overall best performance is achieved using standardization and RNN as model. It can furthermore be seen, that FDIAs based on a scalar are easier to detect than FDIAs based on the distribution of the original measurements. This can be confirmed by computing the average of all $F_1$ scores in Figure 6.7b, resulting in *0.573* and of all $F_1$ scores in Figure 6.7a, which gives *0.831*.




(a) Attack energy based on a scalar.

(b) Attack energy based on the distribution of the original measurements.

Figure 6.7 Averaged $F_1$ score over all tests, grouped by the type of attack energy and the proposed model.

Figure 6.8 shows a comparison of all five preprocessing methods, when grouping all tests into groups corresponding to the five preprocessing methods and computing the average $F_1$ score. It can be seen, that overall standardization performs best.

Although *no preprocessing* results in a solid performance as well, no preprocessing cannot be chosen, because in practice values may span different ranges, therefore resulting

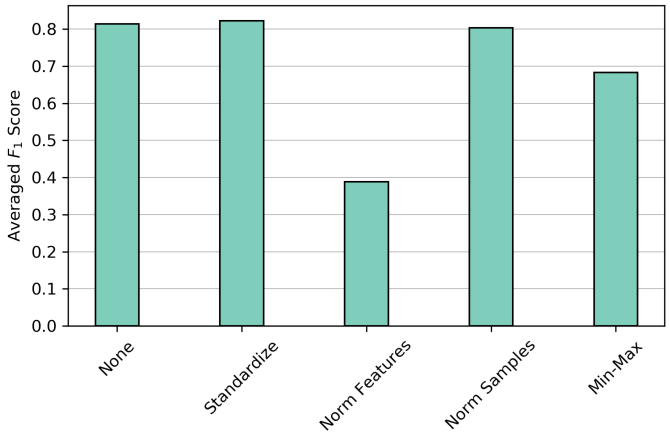



Figure 6.8 Average $F_1$ score of all tests grouped into the five preprocessing methods.

in ranges that the specified model was not trained for. Thus, preprocessing must be applied.

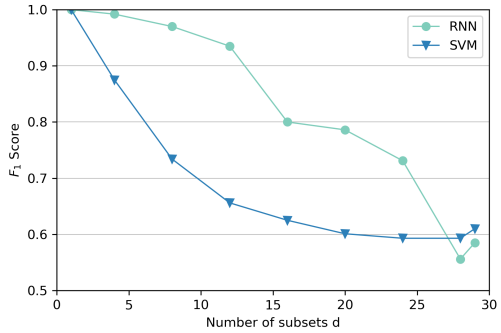
Relating to all previously shown experiments, the following conclusions can be drawn:

- FDIAs constructed based on a scalar are easier to detect than FDIAs constructed based on the distribution of the measurements.
- The best performing preprocessing method for SVM is min-max normalization.
- The best performing preprocessing method for RNN is standardization.
- The overall best performance is achieved using standardization as preprocessing method and RNN as model.
- The FDIAs which are most difficult to detect are the ones constructed with the attack energy defined in $A_4$.

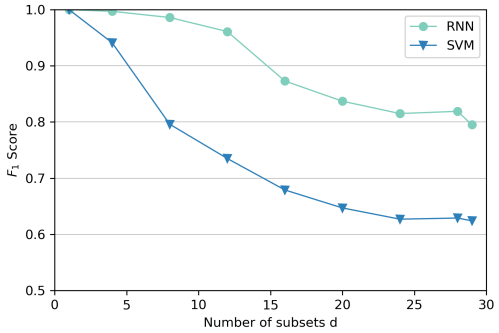
## 6.4 Evalution of Different Numbers of Subsets

In this section, the performance of both SVM and RNN on FDIAs with an attack energy as defined in $A_4$ will be evaluated on different numbers of subsets and attack strengths $P$. As preprocessing methods, the best performing methods are chosen, which is min-max normalization for the SVM and standardization for the RNN.

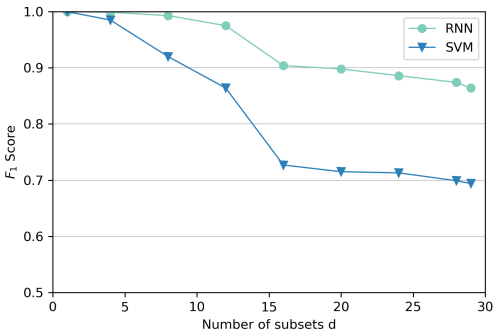
Figure 6.9 shows the $F_1$ score of both RNN and SVM with regard to FDIAs on different numbers of subsets, evaluated on four different attack strengths 0.05, 0.1, 0.3 and 0.5. As numbers of subsets, 1, 4, 8, 12, 16, 20, 24, 28 and 29 are chosen.



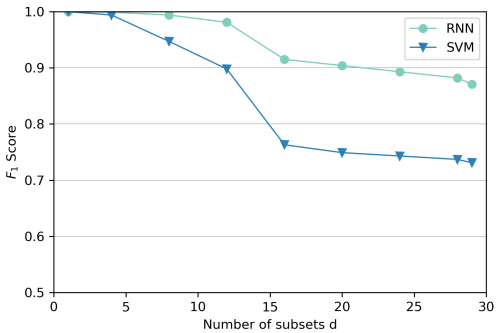

(a) $P = 0.05$.




(b) $P = 0.1$.




(c) $P = 0.3$.

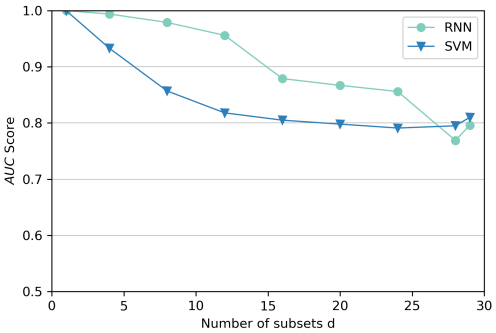



(d) $P = 0.5$.

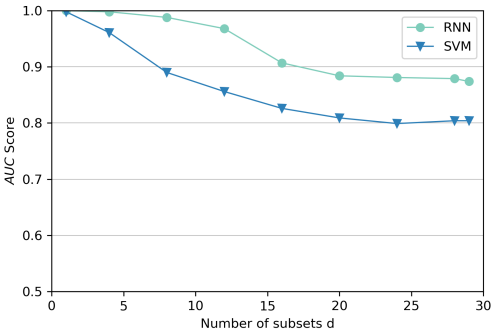
Figure 6.9 Comparison of the $F_1$ score of both RNN and SVM with regard to different numbers of subsets, evaluated on four different types of attack strength $P$.

As can be seen, the RNN outperforms the SVM on all four different attack strengths, except in the case of a high number of subsets ($d = 28$ and $d = 29$) evaluated on the data set containing very few attacks ($P = 0.05$). The good performance of the RNN can be explained by temporal dependency, that is additionally captured by the RNN. In addition, the RNN is trained on more samples than the SVM. It can moreover be seen, that it is easier to detect attacks when the data sets are balanced, such as $P = 0.5$, compared to data sets containing only a few attacks, such as $P = 0.05$. Additionally, as expected, it is harder to detect attacks when the number of subsets increases.

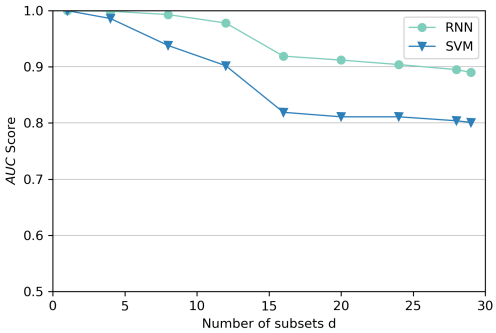
In order to not only depend on one performance metric, Figure 6.10 displays the same evaluation as Figure 6.9, using the AUC score of the ROC as performance metric instead of the $F_1$ score, where a similar result can be seen.



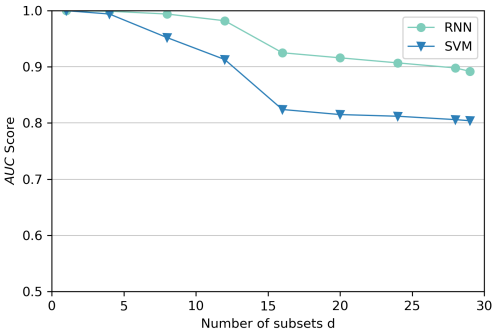

(a) $P = 0.05$.




(b) $P = 0.1$.



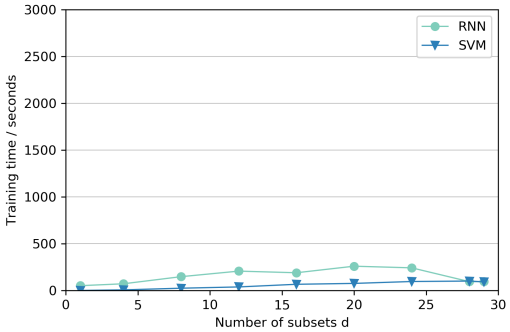

(c) $P = 0.3$.



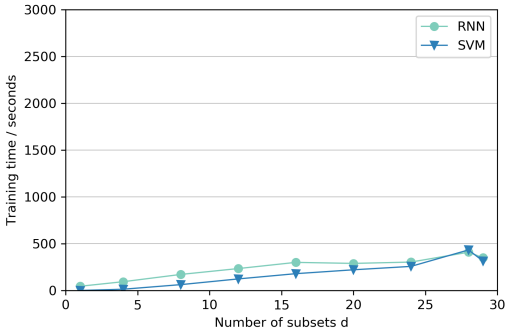

(d) $P = 0.5$.

Figure 6.10 Comparison of the AUC score of both RNN and SVM with regard to different numbers of subsets, evaluated on four different types of attack strength $P$.

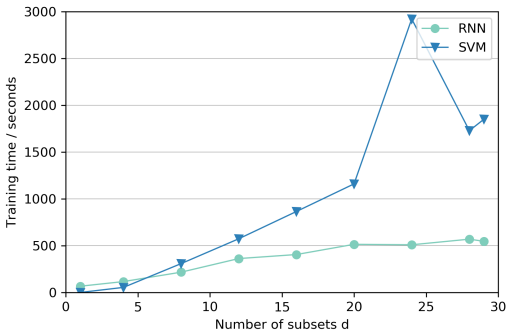
In addition to the $F_1$ score and AUC score, the training time is evaluated and the results are displayed in Figure 6.11. The training of models and when the training is stopped was already investigated in Section 6.1. Although it has to be taken into consideration, that training for SVM and RNN is performed using different implementations and a different hardware, two interesting conclusions can be drawn by looking at the results in Figure 6.11. First, it can be seen that the training time increases proportionally with the number of subsets. Secondly, the training time of the SVM rises rapidly when the data sets get more balanced, whereas the training time of the RNN is constant.



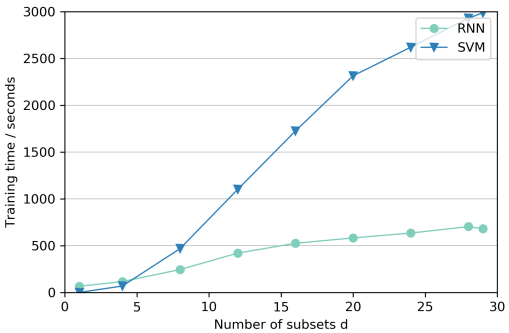

(a) $P = 0.05$.




(b) $P = 0.1$.




(c) $P = 0.3$.




(d) $P = 0.5$.

Figure 6.11 Comparison of the training time of both RNN and SVM with regard to different number of subsets, evaluated on four different types of attack strength $P$.

# Chapter 7

# Conclusions and Outlook

In this chapter, the thesis is summarized and the main achievements are discussed. The objectives specified in Chapter 1 are mapped to their corresponding outcomes. This chapter concludes by discussing possible future work.

## 7.1 Summary

In this thesis, an approach to enhance the current existing research works concerning the detection of FDIAs targeting state estimation in the smart grid, was presented. The overall objective was to investigate the detection of FDIAs targeting a subset level of state estimation variables using machine learning models. Moreover, the need for well documented and reproducible research works in the area of attacks on state estimation, was highlighted.

First, the background regarding cyber-physical systems, smart grids with a specific focus on secure infrastructures, state estimation and attacks on state estimation was given. The need for bad data detection was emphasized. The thesis continued with a survey analyzing almost all of the existing works on defense against attacks targeting state estimation, providing a state-of-the-art overview of the major works in this area up to the time of writing. The survey not only considered defense methods, but also the construction of attacks, different kinds of attacks and incentives for the attacker.

Subsequent to the literature survey, which already mentioned machine learning-based detection approaches, an introduction to machine learning and an explanation of the major concepts was given. Prominent supervised learning algorithms such as SVMs,

NNs and RNNs were investigated, highlighting the advantages and disadvantages of each one.

Moreover, a test framework allowing the generation of attacks based on real-world load data, was proposed. In addition to the generation of attacks, the test framework also featured a completely automated scheduling of tests utilizing both SVM and RNN as supervised learning models. Due to the fact that the test framework was composed with regard to the main principles of object-oriented programming, such as maintainability and code re-usability, additional supervised learning models can be implemented easily. The framework furthermore ensures that all simulations are documented accordingly, resulting in simulations that can be easily reproduced and are well documented.

In the last part of this thesis, experiments were conducted investigating the performance of the proposed models, in specific SVM and RNN, with regard to different types of attacks, attack strengths and numbers of subsets. The major focus was on the detection of attacks targeting a subset level of state estimation variables, which is one of the major contributions of this thesis.

## 7.2 Main Achievements

At the beginning of this thesis, main objectives were identified in Chapter 1. In this section, it is first mentioned how each one of those objectives is connected to their corresponding outcome. Finally, the main achievements are summarized.

- **Extensive literature survey.** An extensive literature survey, considering all major works to the best of the author's knowledge up to the time of writing, was created. Research works were broken down into several distinct categories and tables showing a summary of all the relevant works, were created. Both the construction of different types of attacks, the detection of attacks and the protection against attacks was considered.
- **Test framework.** A test framework was implemented with regard to the concept of object-oriented programming. The framework allows for the generation of data sets containing different types of attacks, the usage of various machine learning-based models and a scheduler for an automated execution of tests.

- **Subset level.** The construction of attacks on a subset level of state estimation variables was both theoretically considered as well as practically implemented as part of the test framework.
- **Machine learning-based detection methods.** As part of machine learning-based detection methods, the supervised learning algorithms Support Vector Machine (SVM) and Recurrent Neural Network (RNN) were theoretically investigated and implemented as part of the test framework. By use of the test framework, the performance of both models was evaluated on various different data sets, including attacks targeting different numbers of subsets with varying attack strengths. It was furthermore shown, that the performance of the machine learning-based detection methods decreased when the number of subsets increased, which was to be expected. Additionally, the proposed machine learning-based detection methods performed better than the current state-of-the-art machine learning-based detection methods.
- **Implementation is open source.** The software for the test framework is provided as a prototypical implementation[1] and made open source under the MIT license model[2].

The solution presented in this thesis addresses two different kinds of stakeholders. Researchers can take advantage of the proposed test framework for an automated conduction of tests, that are well documented at the same time. This ensures that simulations conducted in future research works are reproducible and well documented, increasing the quality of the entire experiments section. In addition, utility companies can use this research to improve the security of their state estimation process and the power grid in general. For example, the grid operator may want to focus on state variables corresponding to buses with a large number of incident branches, because failure of these buses has a high potential of leading to a cascading failure of the entire grid. In a different case, the provider might have alraedy secured a set of basic measurements and wants to specifically monitor state variables not associated with this set of protected measurements.

[1] https://github.com/binarygraka/se-test-framework
[2] https://opensource.org/licenses/MIT

## 7.3 Future Work

This thesis should be viewed as a basis for ongoing future research in the area of the detection of FDIAs targeting state estimation in the smart grid. It must be ensured, that the results in research works are reproducible and that the generation of attack data is clearly documented.

As future work, the concept of subset level detection can be extended to AC state estimation and investigated with regard to this context as well. Moreover, RNNs based on LSTM units can be evaluated with regard to attacks utilizing the temporal dependency in the data, such as replay attacks. Subset level detection can also be evaluated on sparse FDIAs and other types of attacks such as economic attacks. In addition, larger bus systems, such as the 118-bus system, can be investigated with regard to the performance of detecting subset level attacks on state estimation.

The implementation of the supervised machine learning-based algorithm SVM can be implemented in Tensorflow, offering a better 1:1 comparison to the RNN that was already implemented in Tensorflow. Due to the implementation in Tensorflow, the SVM could also be trained on a GPU and the number of samples in the training data set can be increased to the number of samples that are used for the RNN. With regard to the RNN, auxiliary variables and target replication can be investigated in order to improve the detection performance. The prototypical implementation of the test framework will be enhanced by an appropriate documentation.

# Bibliography


[1] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Proceedings of the 47th design automation conference*, ACM, 2010, pp. 731–736.

[2] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, May 2008, pp. 363–369. DOI: 10.1109/ISORC.2008.25.

[3] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid - the new and improved power grid: A survey," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 944–980, 2012, ISSN: 1553-877X. DOI: 10.1109/SURV.2011.101911.00087.

[4] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on cyber security for smart grid communications," *IEEE Communications Surveys and tutorials*, vol. 14, no. 4, pp. 998–1010, 2012.

[5] A. Gómez-Expósito, A. J. Conejo, and C. Canizares, *Electric energy systems: analysis and operation*. CRC press, 2009.

[6] R. Berthier, W. H. Sanders, and H. Khurana, "Intrusion detection for advanced metering infrastructures: Requirements and architectural directions," in *2010 First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 350–355. DOI: 10.1109/SMARTGRID.2010.5622068.

[7] N. Jazdi, "Cyber physical systems in the context of industry 4.0," in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, May 2014, pp. 1–4. DOI: 10.1109/AQTR.2014.6857843.

[8] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, S. Sastry, *et al.*, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, vol. 5, 2009.

[9] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, pp. 161–166, 2011.

[10] A. Abur and A. Gómez-Expósito, *Power system state estimation: theory and implementation*. CRC press, 2004.

[11] B. Zohuri, *Types of Renewable Energy*. Cham: Springer International Publishing, 2018, ISBN: 978-3-319-70721-1. DOI: 10.1007/978-3-319-70721-1_4. [Online]. Available: https://doi.org/10.1007/978-3-319-70721-1_4.

[12] T. Baumeister, "Literature review on smart grid cyber security," *Collaborative Software Development Laboratory at the University of Hawaii*, 2010.

[13] R. R. Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar, "A survey on advanced metering infrastructure," *International Journal of Electrical Power & Energy Systems*, vol. 63, pp. 473–484, 2014.

[14] K. Moslehi and R. Kumar, "A reliability perspective of the smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 57–64, Jun. 2010, ISSN: 1949-3053. DOI: 10.1109/TSG.2010.2046346.

[15] B. Liscouski and W. Elliot, "Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations," *A report to US Department of Energy*, vol. 40, no. 4, p. 86, 2004.

[16] X. Chen, H. Dinh, and B. Wang, "Cascading failures in smart grid - benefits of distributed generation," in *2010 First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 73–78. DOI: 10.1109/SMARTGRID.2010.5622022.

[17] M. Chertkov, F. Pan, and M. G. Stepanov, "Predicting failures in power grids: The case of static overloads," *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 162–172, Mar. 2011, ISSN: 1949-3053. DOI: 10.1109/TSG.2010.2090912.

[18] J. E. Tate and T. J. Overbye, "Line outage detection using phasor angle measurements," *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1644–1652, Sep. 2008, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2008.2004826.

[19] S. Rahman, M. Pipattanasomporn, and Y. Teklu, "Intelligent distributed autonomous power systems (idaps)," in *2007 IEEE Power Engineering Society General Meeting*, Jun. 2007, pp. 1–8. DOI: 10.1109/PES.2007.386043.

[20] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security Privacy*, vol. 7, no. 3, pp. 75–77, May 2009, ISSN: 1540-7993. DOI: 10.1109/MSP.2009.76.

[21] R. Anderson and S. Fuloria, "Who controls the off switch?" In *2010 First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 96–101. DOI: 10.1109/SMARTGRID.2010.5622026.

[22] NIST. (2010). Guidelines for Smart Grid Cyber Security: Vol. 1, Smart Grid Cyber Security Strategy, Architecture, and High-Level Requirements, NISTIR 7628, [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7628r1.pdf (visited on 05/24/2018).

[23] M. Lisovich and S. Wicker, "Privacy concerns in upcoming residential and commercial demand-response systems," *IEEE Proceedings on Power Systems*, vol. 1, no. 1, pp. 1–10, 2008.

[24] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, 13:1–13:33, Jun. 2011, ISSN: 1094-9224. DOI: 10.1145/1952982.1952995. [Online]. Available: http://doi.acm.org/10.1145/1952982.1952995.

[25] Z. Lu, X. Lu, W. Wang, and C. Wang, "Review and evaluation of security threats on the communication networks in the smart grid," in *2010 - MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*, Oct. 2010, pp. 1830–1835. DOI: 10.1109/MILCOM.2010.5679551.

[26] University of Washington. (2018). Power systems test case archive, [Online]. Available: https://www2.ee.washington.edu/research/pstca/ (visited on 05/24/2018).

[27] ——, (1993). 30 bus power flow test case, [Online]. Available: https://www2.ee.washington.edu/research/pstca/pf30/pg_tca30bus.htm (visited on 05/24/2018).

[28] ——, (1973). Ieee common data format, [Online]. Available: https://www2.ee.washington.edu/research/pstca/formats/cdf.txt (visited on 05/24/2018).

[29] A. J. Wood and B. F. Wollenberg, *Power generation, operation, and control*, 3rd ed. John Wiley & Sons, 2014.

[30] V. H. Quintana, A. Simoes-Costa, and A. Mandel, "Power system topological observability using a direct graph-theoretic approach," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-101, no. 3, pp. 617–626, Mar. 1982, ISSN: 0018-9510. DOI: 10.1109/TPAS.1982.317275.

[31] F. C. Schweppe, "Power system static-state estimation - part i, ii & iii," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 1, pp. 130–135, Jan. 1970.

[32] Y. F. Huang, S. Werner, J. Huang, N. Kashyap, and V. Gupta, "State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 33–43, Sep. 2012, ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2187037.

[33] M. Zhou, V. A. Centeno, J. S. Thorp, and A. G. Phadke, "An alternative for including phasor measurements in state estimators," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1930–1937, Sep. 2006, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2006.881112.

[34] A. G. Phadke, J. S. Thorp, and K. J. Karimi, "State estimlatjon with phasor measurements," *IEEE Transactions on Power Systems*, vol. 1, no. 1, pp. 233–238, Feb. 1986, ISSN: 0885-8950. DOI: 10.1109/TPWRS.1986.4334878.

[35] R. E. Larson, W. F. Tinney, L. P. Hajdu, and D. S. Piercy, "State estimation in power systems part ii: Implementation and applications," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 3, pp. 353–363, Mar. 1970, ISSN: 0018-9510. DOI: 10.1109/TPAS.1970.292712.

[36] A. Garcia, A. Monticelli, and P. Abreu, "Fast decoupled state estimation and bad data processing," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-98, no. 5, pp. 1645–1652, Sep. 1979, ISSN: 0018-9510. DOI: 10.1109/TPAS.1979.319482.

[37] J. W. Wang and V. H. Quintana, "A decoupled orthogonal row processing algorithm for power system state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-103, no. 8, pp. 2337–2344, Aug. 1984, ISSN: 0018-9510. DOI: 10.1109/TPAS.1984.318550.

[38] A. Monticelli, *State Estimation in Electric Power Systems: A Generalized Approach*. Springer-Verlag New York, Inc., 1999, ISBN: 9781461372707.

[39] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, Feb. 2011, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2010.2051168.

[40] A. Monticelli and A. Garcia, "Reliable bad data processing for real-time state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-102, no. 5, pp. 1126–1139, May 1983, ISSN: 0018-9510. DOI: 10.1109/TPAS.1983.318053.

[41] H. M. Merrill and F. C. Schweppe, "Bad data suppression in power system static state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-90, no. 6, pp. 2718–2725, Nov. 1971, ISSN: 0018-9510. DOI: 10.1109/TPAS.1971.292925.

[42] L. Mili, T. V. Cutsem, and M. Ribbens-Pavella, "Hypothesis testing identification: A new method for bad data analysis in power system state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-103, no. 11, pp. 3239–3252, Nov. 1984, ISSN: 0018-9510. DOI: 10.1109/TPAS.1984.318561.

[43] L. Mili and T. V. Cutsem, "Implementation of the hypothesis testing identification in power system state estimation," *IEEE Transactions on Power Systems*, vol. 3, no. 3, pp. 887–893, Aug. 1988, ISSN: 0885-8950. DOI: 10.1109/59.14537.

[44] H. He and J. Yan, "Cyber-physical attacks and defences in the smart grid: A survey," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 13–27, 2016.

[45] 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, Jul. 2017, ISSN: 1949-3053. DOI: 10.1109/TSG.2015.2495133.

[46] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, Sep. 2012, ISSN: 1949-3053. DOI: 10.1109/TSG.2012.2195338.

[47] L. Jia, R. J. Thomas, and L. Tong, "On the nonlinearity effects on malicious data attack on power system," in *2012 IEEE Power and Energy Society General Meeting*, Jul. 2012, pp. 1–8. DOI: 10.1109/PESGM.2012.6345685.

[48] M. A. Rahman and H. Mohsenian-Rad, "False data injection attacks against nonlinear state estimation in smart power grids," in *2013 IEEE Power Energy Society General Meeting*, Jul. 2013, pp. 1–5. DOI: 10.1109/PESMG.2013.6672638.

[49] J. Liang, L. Sankar, and O. Kosut, "Vulnerability analysis and consequences of false data injection attack on power system state estimation," *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3864–3872, Sep. 2016, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2015.2504950.

[50] X. Liu and Z. Li, "False data attacks against ac state estimation with incomplete network information," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2239–2248, Sep. 2017, ISSN: 1949-3053. DOI: 10.1109/TSG.2016.2521178.

[51] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*, ACM, New York, Apr. 2009, pp. 21–32.

[52] 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.

[53] G. Dan and H. Sandberg, "Stealth attacks and protection schemes for state estimators in power systems," in *2010 First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 214–219. DOI: 10.1109/SMARTGRID.2010.5622046.

[54] H. Sandberg, A. Teixeira, and K. H. Johansson, "On security indices for state estimators in power networks," in *First Workshop on Secure Control Systems (SCS), Stockholm, 2010*, 2010.

[55] T. T. Kim and H. V. Poor, "Strategic protection against data injection attacks on power grids," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 326–333, Jun. 2011, ISSN: 1949-3053. DOI: 10.1109/TSG.2011.2119336.

[56] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "Malicious data attacks on the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 645–658, Dec. 2011, ISSN: 1949-3053. DOI: 10.1109/TSG.2011.2163807.

[57] K. C. Sou, H. Sandberg, and K. H. Johansson, "Electric power network security analysis via minimum cut relaxation," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec. 2011, pp. 4054–4059. DOI: 10.1109/CDC.2011.6160456.

[58] 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, Jul. 2013, ISSN: 0733-8716. DOI: 10.1109/JSAC.2013.130713.

[59] D. Deka, R. Baldick, and S. Vishwanath, "Optimal hidden scada attacks on power grid: A graph theoretic approach," in *2014 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2014, pp. 36–40. DOI: 10.1109/ICCNC.2014.6785301.

[60] ——, "Data attack on strategic buses in the power grid: Design and protection," in *2014 IEEE PES General Meeting | Conference Exposition*, Jul. 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939058.

[61] J. M. Hendrickx, K. H. Johansson, R. M. Jungers, H. Sandberg, and K. C. Sou, "Efficient computations of a security index for false data attacks in power networks," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3194–3208, Dec. 2014, ISSN: 0018-9286. DOI: 10.1109/TAC.2014.2351625.

[62] 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, Mar. 2014, ISSN: 1949-3053. DOI: 10.1109/TSG.2013.2284438.

[63] 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, no. 3, pp. 717–729, Mar. 2014, ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.92.

[64] J. Hao, R. J. Piechocki, D. Kaleshi, W. H. Chin, and Z. Fan, "Sparse malicious false data injection attacks and defense mechanisms in smart grids," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1–12, Oct. 2015, ISSN: 1551-3203. DOI: 10.1109/TII.2015.2475695.

[65] R. Deng, G. Xiao, and R. Lu, "Defending against false data injection attacks on power system state estimation," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 198–207, Feb. 2017, ISSN: 1551-3203. DOI: 10.1109/TII.2015.2470218.

[66] Q. Yang, D. An, R. Min, W. Yu, X. Yang, and W. Zhao, "On optimal pmu placement-based defense against data integrity attacks in smart grid," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1735–1750, Jul. 2017, ISSN: 1556-6013. DOI: 10.1109/TIFS.2017.2686367.

[67] M. Esmalifalak, H. Nguyen, R. Zheng, and Z. Han, "Stealth false data injection using independent component analysis in smart grid," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2011, pp. 244–248. DOI: 10.1109/SmartGridComm.2011.6102326.

[68] M. A. Rahman and H. Mohsenian-Rad, "False data injection attacks with incomplete information against smart power grids," in *2012 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2012, pp. 3153–3158. DOI: 10.1109/GLOCOM.2012.6503599.

[69] V. Kekatos, G. B. Giannakis, and R. Baldick, "Grid topology identification using electricity prices," in *2014 IEEE PES General Meeting | Conference Exposition*, Jul. 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939474.

[70] X. Liu, Z. Bao, D. Lu, and Z. Li, "Modeling of local false data injection attacks with reduced network information," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1686–1696, Jul. 2015, ISSN: 1949-3053. DOI: 10.1109/TSG.2015.2394358.

[71] Z. H. Yu and W. L. Chin, "Blind false data injection attack using pca approximation method in smart grid," *IEEE Transactions on Smart Grid*, vol. 6, no. 3, pp. 1219–1226, May 2015, ISSN: 1949-3053. DOI: 10.1109/TSG.2014.2382714.

[72] J. Kim and L. Tong, "On topology attack of a smart grid: Undetectable attacks and countermeasures," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1294–1305, Jul. 2013, ISSN: 0733-8716. DOI: 10.1109/JSAC.2013.130712.

[73] L. Jia, J. Kim, R. J. Thomas, and L. Tong, "Impact of data quality on real-time locational marginal price," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 627–636, Mar. 2014, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2013.2286992.

[74] M. A. Rahman, E. Al-Shaer, and R. Kavasseri, "Impact analysis of topology poisoning attacks on economic operation of the smart power grid," in *2014 IEEE 34th International Conference on Distributed Computing Systems*, Jun. 2014, pp. 649–659. DOI: 10.1109/ICDCS.2014.72.

[75] R. Deng, P. Zhuang, and H. Liang, "Ccpa: Coordinated cyber-physical attacks and countermeasures in smart grid," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2420–2430, Sep. 2017, ISSN: 1949-3053. DOI: 10.1109/TSG.2017.2702125.

[76] Y. Yuan, Z. Li, and K. Ren, "Modeling load redistribution attacks in power systems," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 382–390, Jun. 2011, ISSN: 1949-3053. DOI: 10.1109/TSG.2011.2123925.

[77] ——, "Quantitative analysis of load redistribution attacks in power systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1731–1738, Sep. 2012, ISSN: 1045-9219. DOI: 10.1109/TPDS.2012.58.

[78] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *2010 First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 226–231. DOI: 10.1109/SMARTGRID.2010.5622048.

[79] ——, "Integrity data attacks in power market operations," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 659–666, Dec. 2011, ISSN: 1949-3053. DOI: 10.1109/TSG.2011.2161892.

[80] L. Jia, R. J. Thomas, and L. Tong, "Malicious data attack on real-time electricity market," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5952–5955. DOI: 10.1109/ICASSP.2011.5947717.

[81] D.-H. Choi and L. Xie, "Impact analysis of locational marginal price subject to power system topology errors," in *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2013, pp. 55–60. DOI: 10.1109/SmartGridComm.2013.6687933.

[82] M. Esmalifalak, G. Shi, Z. Han, and L. Song, "Bad data injection attack and defense in electricity market using game theory study," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 160–169, Mar. 2013, ISSN: 1949-3053. DOI: 10.1109/TSG.2012.2224391.

[83] J. Yan, Y. Tang, B. Tang, H. He, and Y. Sun, "Power grid resilience against false data injection attacks," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, Jul. 2016, pp. 1–5. DOI: 10.1109/PESGM.2016.7741850.

[84] J. W. Kang, I. Y. Joo, and D. H. Choi, "False data injection attacks on contingency analysis: Attack strategies and impact assessment," *IEEE Access*, vol. 6, pp. 8841–8851, 2018. DOI: 10.1109/ACCESS.2018.2801861.

[85] J. Kim, L. Tong, and R. J. Thomas, "Data framing attack on state estimation," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 7, pp. 1460–1470, Jul. 2014, ISSN: 0733-8716. DOI: 10.1109/JSAC.2014.2332032.

[86] A. Gul and S. D. Wolthusen, "Measurement re-ordering attacks on power system state estimation," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Sep. 2017, pp. 1–6. DOI: 10.1109/ISGTEurope.2017.8260145.

[87] D. Deka, R. Baldick, and S. Vishwanath, "Optimal data attacks on power grids: Leveraging detection measurement jamming," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2015, pp. 392–397. DOI: 10.1109/SmartGridComm.2015.7436332.

[88] ——, "Jamming aided generalized data attacks: Exposing vulnerabilities in secure estimation," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, Jan. 2016, pp. 2556–2565. DOI: 10.1109/HICSS.2016.319.

[89] L. Wei, A. I. Sarwat, W. Saad, and S. Biswas, "Stochastic games for power grid protection against coordinated cyber-physical attacks," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 684–694, Mar. 2018, ISSN: 1949-3053. DOI: 10.1109/TSG.2016.2561266.

[90] 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, Sep. 2015, ISSN: 1949-3053. DOI: 10.1109/TSG.2015.2388545.

[91] S. Bi and Y. J. Zhang, "Defending mechanisms against false-data injection attacks in the power system state estimation," in *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2011, pp. 1162–1167. DOI: 10.1109/GLOCOMW.2011.6162362.

[92] ——, "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, May 2014, ISSN: 1949-3053. DOI: 10.1109/TSG.2013.2294966.

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

[94] C. Wickramaarachchi, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Improved protection scheme for data attack on strategic buses in the smart grid," in *2016 IEEE Conference on Technologies for Sustainability (SusTech)*, Oct. 2016, pp. 96–101. DOI: 10.1109/SusTech.2016.7897149.

[95] M. Talebi, C. Li, and Z. Qu, "Enhanced protection against false data injection by dynamically changing information structure of microgrids," in *2012 IEEE 7th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, Jun. 2012, pp. 393–396. DOI: 10.1109/SAM.2012.6250520.

[96] B. Milosevic and M. Begovic, "Voltage-stability protection and control using a wide-area network of phasor measurements," *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 121–127, Feb. 2003, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2002.805018.

[97] A. A. A. Esmin, G. Lambert-Torres, and A. C. Z. de Souza, "A hybrid particle swarm optimization applied to loss power minimization," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 859–866, May 2005, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2005.846049.

[98] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "On malicious data attacks on power system state estimation," in *45th International Universities Power Engineering Conference UPEC2010*, Aug. 2010, pp. 1–6.

[99] Y. Huang, H. Li, K. A. Campbell, and Z. Han, "Defending false data injection attack on smart grid network using adaptive cusum test," in *2011 45th Annual Conference on Information Sciences and Systems*, Mar. 2011, pp. 1–6. DOI: 10.1109/CISS.2011.5766111.

[100] S. Bhattarai, L. Ge, and W. Yu, "A novel architecture against false data injection attacks in smart grid," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 907–911. DOI: 10.1109/ICC.2012.6364511.

[101] T. Liu, Y. Gu, D. Wang, Y. Gui, and X. Guan, "A novel method to detect bad data injection attack in smart grid," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 49–54. DOI: 10.1109/INFCOMW.2013.6562907.

[102] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 4, pp. 370–379, Dec. 2014, ISSN: 2325-5870. DOI: 10.1109/TCNS.2014.2357531.

[103] P. Y. Chen, S. Yang, J. A. McCann, J. Lin, and X. Yang, "Detection of false data injection attacks in smart-grid systems," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 206–213, Feb. 2015, ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7045410.

[104] S. Li, Y. Yılmaz, and X. Wang, "Quickest detection of false data injection attack in wide-area smart grids," *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 2725–2735, Nov. 2015, ISSN: 1949-3053. DOI: 10.1109/TSG.2014.2374577.

[105] W. Yu, D. Griffith, L. Ge, S. Bhattarai, and N. Golmie, "An integrated detection system against false data injection attacks in the smart grid," *Security and Communication Networks*, vol. 8, no. 2, pp. 91–109, 2015.

[106] F. Pasqualetti, R. Carli, and F. Bullo, "A distributed method for state estimation and false data detection in power networks," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2011, pp. 469–474. DOI: 10.1109/SmartGridComm.2011.6102368.

[107] H. Sedghi and E. Jonckheere, "Statistical structure learning of smart grid for detection of false data injection," in *2013 IEEE Power Energy Society General Meeting*, Jul. 2013, pp. 1–5. DOI: 10.1109/PESMG.2013.6672176.

[108] 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)*, Nov. 2012, pp. 312–317. DOI: 10.1109/SmartGridComm.2012.6486002.

[109] Y. Chakhchoukh, S. Liu, M. Sugiyama, and H. Ishii, "Statistical outlier detection for diagnosis of cyber attacks in power state estimation," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, Jul. 2016, pp. 1–5. DOI: 10.1109/PESGM.2016.7741572.

[110] J. Yan, B. Tang, and H. He, "Detection of false data attacks in smart grid with supervised learning," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 1395–1402. DOI: 10.1109/IJCNN.2016.7727361.

[111] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, Sep. 2017, ISSN: 1932-8184. DOI: 10.1109/JSYST.2014.2341597.

[112] S. A. Foroutan and F. R. Salmasi, "Detection of false data injection attacks against state estimation in smart grids based on a mixture gaussian distribution learning method," *IET Cyber-Physical Systems: Theory Applications*, vol. 2, no. 4, pp. 161–171, 2017. DOI: 10.1049/iet-cps.2017.0013.

[113] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2505–2516, Sep. 2017, ISSN: 1949-3053. DOI: 10.1109/TSG.2017.2703842.

[114] Y. Wang, M. M. Amin, J. Fu, and H. B. Moussa, "A novel data analytical approach for false data injection cyber-physical attack mitigation in smart grids," *IEEE Access*, vol. 5, pp. 26 022–26 033, 2017. DOI: 10.1109/ACCESS.2017.2769099.

[115] A. Ayad, H. E. Z. Farag, A. Youssef, and E. F. E. El-Saadany, "Detection of false data injection attacks in smart grids using recurrent neural networks," 2018, in press.

[116] H. Wang, J. Ruan, G. Wang, B. Zhou, Y. Liu, X. Fu, and J. C. Peng, "Deep learning based interval state estimation of ac smart grids against sparse cyber attacks," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018, ISSN: 1551-3203. DOI: 10.1109/TII.2018.2804669.

[117] 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, Aug. 2016, ISSN: 2162-237X. DOI: 10.1109/TNNLS.2015.2404803.

[118] J. J. Q. Yu, Y. Hou, and V. O. K. Li, "Online false data injection attack detection with wavelet transform and deep neural networks," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018, ISSN: 1551-3203. DOI: 10.1109/TII.2018.2825243.

[119] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[120] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed., ser. Adaptive computation and machine learning. MIT Press, 2010, ISBN: 9780262012430. [Online]. Available: https://books.google.com/books?id=4j9GAQAAIAAJ.

[121] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006, ISBN: 0387310738.

[122] W. H. Jefferys and J. O. Berger, "Ockham's razor and bayesian analysis," *American Scientist*, vol. 80, no. 1, pp. 64–72, 1992, ISSN: 00030996. [Online]. Available: http://www.jstor.org/stable/29774559.

[123] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[124] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, ISSN: 1522-9602. DOI: 10.1007/BF02478259. [Online]. Available: https://doi.org/10.1007/BF02478259.

[125] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015. arXiv: 1506.00019. [Online]. Available: http://arxiv.org/abs/1506.00019.

[126] M. Boden, "A guide to recurrent neural networks and backpropagation," *the Dallas project*, 2002.

[127] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[128] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[129] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[130] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information retrieval*, vol. 1, no. 1-2, pp. 69–90, 1999.

[131] New York Independent System Operator. (2018). Load data, [Online]. Available: http://www.nyiso.com/public/markets_operations/market_data/load_data/index.jsp (visited on 05/24/2018).

[132] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. E. Lorensen, *et al.*, *Object-oriented modeling and design*, 1. Prentice-hall Englewood Cliffs, NJ, 1991, vol. 199.

[133] Power Systems Engineering Research Center. (2016). Guidelines for Smart Grid Cyber Security: Vol. 1, Smart Grid Cyber Security Strategy, Architecture, and High-Level Requirements, NISTIR 7628, [Online]. Available: http://www.pserc.cornell.edu/matpower/MATPOWER-manual.pdf (visited on 07/01/2018).

[134] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.