Hybrid deep neural network anomaly detection system for SCADA networks

Raogo Kabore

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité: *Informatique*

Par
Raogo KABORÉ

Hybrid Deep Neural Network Anomaly Detection System for SCADA Networks

Thèse présentée et soutenue à IMT Atlantique, Brest, le 11 juin 2020
Unité de recherche :IRIS / Lab-STICC UMR CNRS 6285 - UBL
Thèse N°: 2020IMTA0190

Rapporteurs avant soutenance :

Martine COLLARD	Professeur, Université des Antilles, Guadeloupe
Bouabid EL OUAHIDI	Professeur, Université Mohamed V de Rabat

Composition du Jury :

Président	: Philippe LENCA	Professeur, IMT Atlantique, Brest
Examinateurs	: Yvon KERMARREC	Professeur, IMT Atlantique, Brest
	Martine COLLARD	Professeur, Université des Antilles, Guadeloupe
	Bouabid EL OUAHIDI	Professeur, Université Mohamed V de Rabat

Directeur de thèse	: Yvon KERMARREC	Professeur, IMT Atlantique, Brest
Co-encadrant de thèse	: Philippe LENCA	Professeur, IMT Atlantique, Brest

To my father and mother

To my beloved wife, for all the sacrifices and support

To my dear children

To my late twin Kalga

To my sister and little mom Minata

Aknowlegments

I would like to acknowledge all the persons who helped me in the course of this Thesis. First of all, my thoughts goes to Yvon Kermarrec, my Ph.D Director. Thank you for accepting me as a PhD student, trusting me during this Thesis, sharing ideas with me, and encouraging me during hard times. Many thanks to Philippe Lenca, the Director of the LUSSI Department, and co-Director of this Thesis. Thank you for your advice, encouragement, and for the warm welcome in the LUSSI Department.
Special thanks to Laurent Nana and Eric Saux, the members of my individual follow-up committee, or comité de suivi individual (CSI) in french who were evaluating my work each year. Thank you for your relevant remarks and advice during those three CSIs which help me each time reshape my research.
Thanks to Adama Konaté, the General Manager of our engineering school ESATIC in Côte d'Ivoire for establishing the partnership with IMT Atlantique through which we had the opportunity to do this Thesis. I also want to thank the Education Director of ESATIC and head of Department, Etienne Soro for the flexibility in the schedule that makes possible our Thesis en part-time away from the school. Thanks to all my colleagues of ESATIC.
My thoughts also goes to the LUSSI Department Assistant Ghislaine Le Gall. Thank you so much for your devotion to make our stay easy and for the support for administrative tasks.
I would also like to thank Philippe Tanguy and his wife for their friendship. Many thanks to Sebastien Bigaret of LUSSI and Emmanuel Braux of the DISI Department, who spared no effort in helping me with technical issues surrounding my development environment. Thanks to all the colleagues of the LUSSI Department, for your support and advice. Thanks to Daniel Bourget for accepting me in his Specialized Master in Web Technology and Cyber Security (TWCS) which gives me the opportunity of this Thesis. Thanks to my colleague Zakariya Kamagaté with whom I started the adventure of this Thesis, and shared the office at LUSSI. Thanks for all the ideas and the friendship during all those years.
Finally, I would like to thank all the persons involved at any level in the fulfillment of this Thesis.

Table of Contents

List of Tables

List of Figures

General Introduction

Context

SCADA systems

SCADA which stands for Supervisory Control And Data Acquisition, are Industrial Control Systems (ICS) used for centralized data acquisition and control of geographically dispersed assets. Those systems are used in water distribution, wastewater treatment, power transmission and distribution, oil and gas pipelines, public transportation systems, etc. [145]. The integration of SCADA systems to the management of industrial systems not only helps improve performance, but also reduces operating costs [21].

Historically, no consideration was given to security at the time of the design of SCADA systems, but engineers were more focused on practical aspects such as availability, reliability, and performance of physical processes [91]. Designers relied upon two forms of protection i.e. air gap [22] and security through obscurity. The former was based on the fact that SCADA networks are physically isolated from other networks making any attack difficult, while the latter relied on the presumption that information about SCADA systems are not available to public, thereby making them secured [94].

Attacks on SCADA systems

Nowadays, modern SCADA systems use Commercial-Off-The-Shelf (COTS) hardware, software, standard communication technologies such as TCP/IP and ETH-

ERNET or Wireless protocols. Moreover, today's SCADA networks are interconnected to corporate networks and to the Internet for diverse reasons such as management, system administration, maintenance etc. [23] [134]. Those shifts in SCADA networks though allowing easy management and reduction of costs, expose them to cyber-attacks [125] [21].
The consequences of an attack on SCADA systems can be loss of production, financial losses, environmental disasters and even loss of human lives [154] [169].
SCADA networks have already been the target of attacks such as those of the Maroochy Water System in Australia [142], the Davis-Besse nuclear power plant in Ohio (USA), the Iranian uranium enrichment centrifuges with Stuxnet [48] [29], power plants in Ukraine [102] and Vermont in the USA [129].

SCADA Intrusion Detection Systems (IDS)

SCADA networks are different from traditional IT ones, and those specificities must be taken into account in approaches to design security solutions for SCADA systems. Unlike in traditional IT, software patching and frequent updates are not well suited for control systems. Those systems also require high availability, have large amount of legacy systems and usually have static topology and regular communication patterns [23] [170] [30]. Moreover, the embedded systems used to implement industrial automation systems also suffer from restrictions such as memory and processing limitations, lack of robustness and software implementation and configuration issues [46].
In order to protect SCADA networks, [26] proposes many SCADA-specific security standards such as firewall deployment, message monitoring, protocol-based solutions, cryptographic key management, anti-viruses and software patches. Consistent security policy, well-designed network architecture, system hardening, two-factor password and data encryption for remote connections are also useful to secure SCADA systems [35]. Moreover, [125] propose authentication of communication partners, use of secure protocols and Virtual Private Networks (VPN).

Despite the combination of all those security measures, there is no zero-risk in information systems security, i.e., an attack could succeed in the network. Intrusion detection systems (IDS) and anomaly detection systems are security solutions that plays a significant role in information security, as they help detect successful intrusions or anomalous events in a network. [56] advocates that IDS should be used along with other security mechanisms such as firewalls, vulnerability scanners, security policy verifiers to ensure an optimal industrial control network security. But the specific nature of SCADA systems requires specific approaches for SCADA intrusion/anomaly detection systems.

There are mainly three kinds of intrusion detection approaches : signature or misuse detection, anomaly detection and model-based or specification detection [8] [170]. The signature detection (also known as misuse detection) matches the traffic to a know signature. On the other hand, anomaly detection learns the "normal" behavior of the system and tries to detect abnormalities in traffic i.e. traffic deviating from the “normal” behavior. The main drawbacks of anomaly detection is the low detection rate and high false alarm. But anomaly detection based IDS can detect new or zero-day attacks. The model-based approach uses rules to create a model of what is allowed and raises alerts when the observed behavior is not matching the rules. Although sounding promising, it is hard to entirely model a system [8] [169]. Therefore, anomaly detection is an important defense mechanism to protect SCADA systems.

Objective and Research Questions

In recent years, Deep Learning [59] which is a sub-field of machine learning became a hot topic among researchers as this approach is successfully applied to domains like image classification, video and natural language processing (NLP). In the literature, there is more and more attempts to use deep learning for networks anomaly detection [6] [44] [50] [51] [52] [67] [80] [84] [85] [86]. But, very few unsupervised deep neural network approaches have been used for anomaly detection in the specific domain of SCADA networks.

Anomaly-based intrusion detection systems are either supervised or unsupervised [99]. In supervised methods, training data are labeled "normal" or "abnormal" by the domain expert and the system is trained to discriminate between "normal" or "abnormal" observations, so that new observations could be classify to "normal" or "abnormal" classes. In unsupervised approaches there is no labeled data. A baseline distribution of "normal" behaviors is modeled, so that the system could detect observations that show significant difference from the "normal" ones [99] [157].
But the labeling of huge datasets by human experts is costly, time consuming and error-prone.
Deep learning techniques can use unsupervised strategies to automatically learn hierarchical representations in deep architectures for classification purpose [20] [59] [92] [97] [104] [105] [113] [130] [132].

However, training deep learning models remains quite challenging because of the huge amount of time required for the training process.

My research objective in this thesis is to propose an accurate anomaly detection system in terms of detection rate and false positive rate, and efficient in terms of processing time in SCADA networks, using an unsupervised deep feature learning approach. In order to reach this objective, we must address the following research questions :

RQ 1 : How can we design a deep learning based approach for unsupervised feature learning in SCADA systems?

To answer this question, we have to go through three sub-questions :

RQ 1.1 : What is the state of the art of deep learning based unsupervised feature learning in SCADA systems?

This question implies conducting a review of the state of the art in using deep learning unsupervised feature learning for anomaly detection in SCADA systems.

The second sub-question is :

RQ 1.2 : What are the characteristics of the SCADA dataset used ?

The answer to this question will help us have a clear understanding of the SCADA dataset used i.e. the nature of the records and the different types of attacks it contains.

The last sub-question is :

RQ 1.3 : How can we design a deep learning based unsupervised feature learning framework that will learn important features from SCADA data ?

To answer this question, we will show the process of designing a deep neural network for unsupervised feature learning in SCADA systems, i.e, how to determine the parameters, the activation functions of the layers, the loss function to be used, the various hyper-parameters such as the learning rate, batch size, regularization, etc., and the training method of the model.

The second research question to answer for our objective is :

RQ 2 : How can we design a deep learning based framework for efficient and accurate anomaly detection in SCADA systems?

This research question is divided into three sub-questions as well :

RQ 2.1 : How do we build a deep learning general framework for anomaly detection in SCADA systems?

To answer this question, we will build the different components of the SCADA anomaly detection system, i.e. the different parts of the pre-processing engine as well as the different parts of the anomaly detection engine.

The next sub-question is :

RQ 2.2 : Using the unsupervised deep feature learning approach of the previous research question, can we build an efficient and accurate classifier for anomaly detection in SCADA networks?

To answer this question, we will show how to build the actual classifier for anomaly detection, based on the unsupervised deep feature learner.

And the last sub-question of the second research question is :

RQ 2.3 : How can we boost the computational efficiency for deep neural network based anomaly detection system in SCADA networks?

As training deep learning models is computationally intensive, through this question, we will propose a distributed version of the anomaly detection model

in order to train the proposed deep learning model faster.

Contributions

In the course of this thesis, we did five main contributions :

Contribution 1
Our first contribution is a review of deep neural network-based SCADA anomaly detection systems using feature learning approaches.

Contribution 2
For the second contribution, we proposed a stacked sparse denoising autoencoder architecture as feature learner for SCADA networks. This framework is able to automatically learn the salient features of the SCADA data that will later be used for classification purpose.

Contribution 3
A framework for an hybrid deep neural network SCADA anomaly detection system which has a data pre-processing engine and an anomaly detection engine is proposed as the third contribution. In the pre-processing engine, data are normalized, balanced and one-hot encoded. The anomaly detection engine has an unsupervised feature learning module to which is added a supervised classifier.

Contribution 4
To speed-up the training process of the hybrid deep neural network SCADA anomaly detection system, we proposed as a fourth contribution a distributed version which uses a parameter server and worker nodes. The parameter server stores the model weights and distributes it to the worker nodes were all the workload happens. Each of the worker node is responsible for computing the gradient during the back-propagation training algorithm. The gradient calculated by each worker node is sent back to the parameter server which updates the weights and propagates it back to worker nodes.

Contribution 5
Finally, our fifth contribution is an implementation of the hybrid deep neural network SCADA anomaly detection system with TensorFlow framework proved that this approach gives in overall better results in terms of detection rate and false alarm rate compared to baseline algorithms such as Decision Tree, Naïve Bayes and Random Forest.
For the distributed version, implementing the proposed approach using the Distributed TensorFlow framework on a Hadoop Cluster, reduces the training time significantly, compared to single machine implementation.

Thesis Organization

The manuscript is organized into four parts (Figure 1) i.e. the general introduction, the state of the art, the contributions and finally, the general conclusion. After the general introduction, the state of the art has two chapters i.e. chapter 1 and 2. Chapter 1 is an overview of SCADA systems architecture, their vulnerabilities, the existing attacks against them, and a review of existing SCADA intrusion detection systems. Chapter 2 is dedicated to explain deep learning techniques and architectures to the reader. The third part of the manuscript is the contributions organized into four chapters (3 through 6). In the chapter 3, we propose a review of SCADA anomaly detection systems using deep feature learning approach. This chapter is followed by the design of an unsupervised deep neural network feature learning framework for SCADA systems in chapter 4. The previously built framework is used in the overall design of an hybrid deep neural network SCADA Anomaly Detection System in chapter 5. The proposed approach implementation and results are given in chapter 6 before a general conclusion.

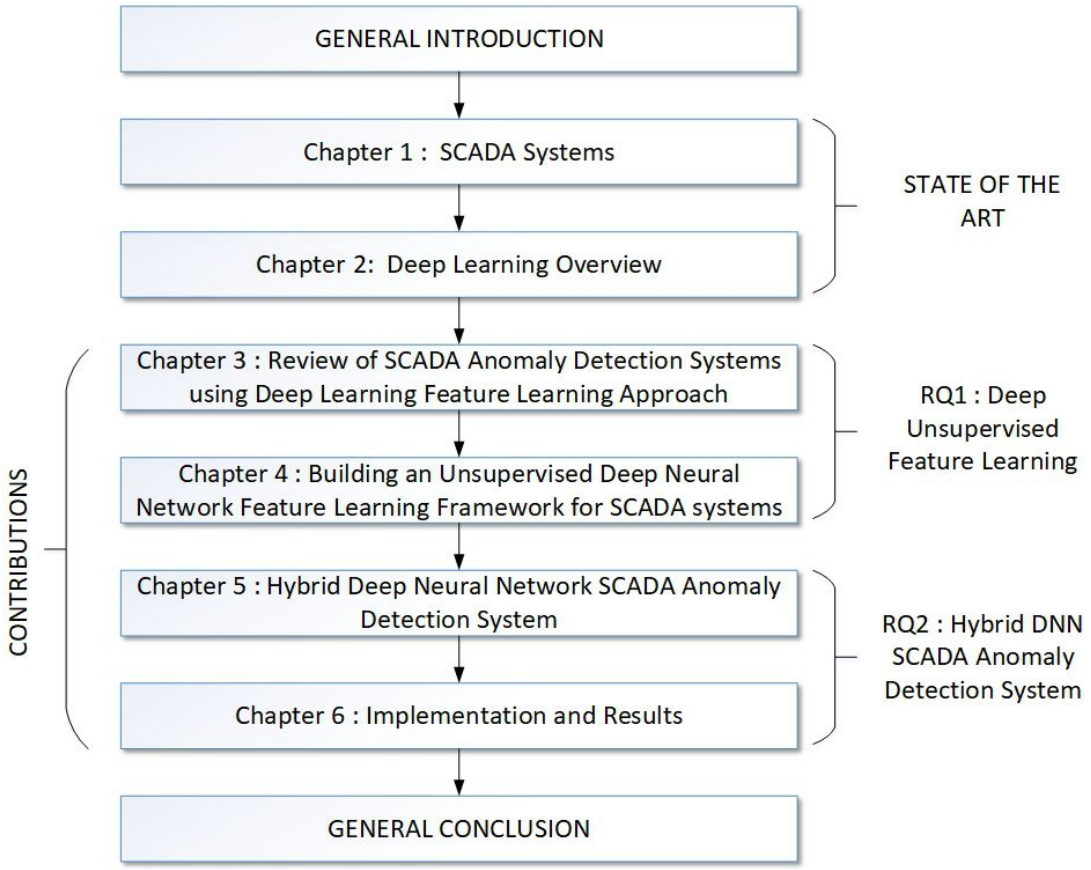

Figure 1 – Thesis Organization

Part I

State of the Art

Chapter 1

SCADA Systems

1.1 Introduction

In order to monitor and control industrial systems such as smart grids, nuclear power plants, gas pipelines, manufacturing plants, etc., experts designed industrial control systems also known as SCADA systems. However, at the time of their design, they were more focused on practical aspects such as availability, reliability, performance, longevity of physical processes [91]. Moreover, as time evolves, not only Commercial-Off-The-Shelf (COTS) hardware and software as well as standard protocols like Ethernet and TCP/IP are used within SCADA systems, but they also get interconnected with corporate networks and the Internet.

In the present work, we show the different vulnerabilities of SCADA networks , and how those vulnerabilities could be exploited by threat actors to compromise them. Although many security approaches have been proposed to secure SCADA systems, we show that a special attention should be paid to intrusion/anomaly detection systems to secure those networks.

After an overview of SCADA systems architectures, we highlight different vulnerabilities within them that could be exploited by attackers. In the following section, some documented attacks against SCADA networks are presented. Thereafter, different solutions to secure SCADA systems are proposed, and a

full section is dedicated to existing SCADA intrusion detection systems before concluding.

1.2 SCADA Systems Architecture

Industrial Control System (ICS) is a more general term used to designate SCADA systems or Distributed Control Systems (DCS). In the literature, those terms are used interchangeably [94] [28] [91]. We would henceforth use the term SCADA to designate any industrial control system in the remainder of this dissertation.

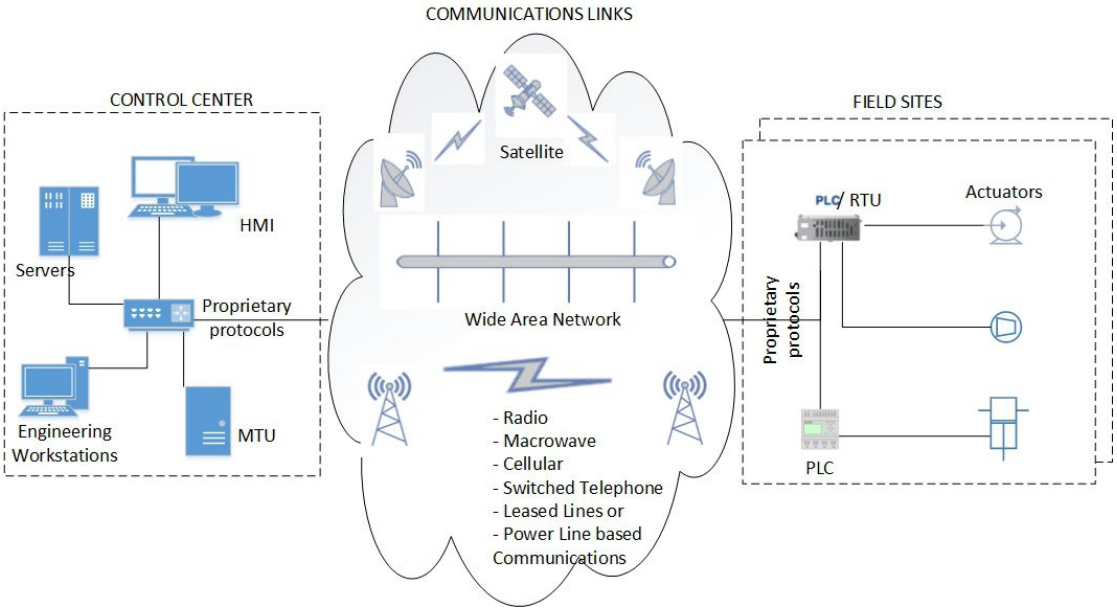

Figure 1.1 – SCADA system general layout

1.2.1 Components of a SCADA system

A typical SCADA system (Figure 1.1) has three parts that are the control center, the communication links and one or more distributed field sites [145].
The control center is the part of a SCADA network where the Master Terminal Unit (MTU) also called SCADA server, the engineering workstations, the Human Machine Interfaces (HMIs), and the data historian are located. The Master Terminal Unit (MTU) is the key component of the control center. It is used to store and

process information from the Remote Terminal Units (RTUs). The monitoring and control of the entire SCADA network is done by human operators thanks to the HMIs, while the engineering workstations are used by engineers for maintenance and configuration purpose. Finally, all the information collected on the SCADA networks are store on a database called the data historian for further processing.
The second component of a SCADA network are the communication links which are used to transfer the information between the MTU and the RTUs or PLCs. The Communication links are cable, fiber, radio, telephone lines or satellite.
The last component of SCADA networks are the field sites where we find components such as Remote Terminal Units (RTU), Programmable Logic Controller (PLC), actuators (motors, valves, fans, etc.), and sensors that measure the values of the physical phenomena (pressure, oil level, temperature, etc.). PLCs defined by the IEC 61131 are special form of microprocessor-based controllers that use programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting, and arithmetic in order to control machines and processes [18]. An RTU typically resides in a substation or some remote location. The role of RTUs is to monitor field parameters and transmit data back to an MTU, a centrally located PLC, or directly to an HMI [91].

1.2.2 SCADA systems protocols

Most SCADA systems use proprietary protocol (such as Honeywell CDA, General Electric SRTP or Siemens S7, etc.) or non-proprietary and/or licensed protocols such as OPC, Modbus, DNP3, ICCP, CIP, PROFIBUS, etc. Originally designed for serial communications, many of these protocols have been adapted to operate with Ethernet, TCP/IP or UDP Protocols [91].
Fieldbus protocols as indicated by the name are used on field sites. They are defined by standard IEC 61158 as a set of protocols deployed to connect process-connected devices (e.g. sensors) to basic control devices (e.g. programmable logic controller or PLC), and control devices to supervisory systems (e.g. ICS

server, human–machine interface or HMI, historian) [91]. In the control center, the main application-level protocols used is SCADA systems are Modbus [112], DNP3 [47] [109] and IEC 60870-5 [81].

. The Modbus transmission protocol was developed by Gould Modicon [112] (now Schneider) for process control systems.

In addition to the standard Modbus protocol, there are other versions of the protocol such as Modbus Plus, Modbus TCP. The standard Modbus is a serial protocol which operates on the master/slave principle, a master for up to 247 slaves. Only the master initiates the transaction. Modbus Plus is a peer-to-peer protocol which runs at 1 mbs. The Modbus Plus protocol specifies the software layer as well as the hardware layer. And finally, the Modbus TCP/IP is simply the Modbus RTU protocol with a TCP interface that runs on Ethernet.

On the standard Mobdus which is the most common implementation, the transactions use request/response scheme where only one slave is requested, or of broadcast/unanswered type where all the slaves are polled [33]. A Modbus message frame encompasses an address field of 1 byte, a function field of 1 byte, a data field of variable length and an error check filed of 2 bytes. Each request frame contains a function code that defines the action desired by the controller. The meaning of the query data field depends on the code of the specified function.

The second control protocol is the Distributed Network Protocol Version 3.3 (DNP3).

DNP3 is a telecommunication standard that defines communications between master stations, Remote Telemetry Units (RTUs) and other Intelligent Electronic Devices (IEDs). DNP3 uses the Enhanced Performance Architecture (EPA) three layer architecture (physical, data link, and application) to which a pseudo-transport layer is added. The DNP3 protocol therefore uses the Physical, Data link, Pseudo-transport and Application layers. As the Modbus protocol, the DNP3 protocol uses function codes in its message frames.

In a move for a standardized protocol, The International Electro-technical Commission or IEC came-up with IEC 60870-5 that refers to a set of standards produced to provide an open standard for the transmission of SCADA telemetry information and controls. Like DNP3, IEC 60870-5 is based on the three layers of

the Enhanced Performance Architecture or EPA model for data communication. Modbus is lightweight compared to DNP3 and IEC 60870-5 which are more complex and offers more interoperability compared to Modbus.

1.3 SCADA Systems vulnerabilities and attacks

1.3.1 SCADA Systems vulnerabilities

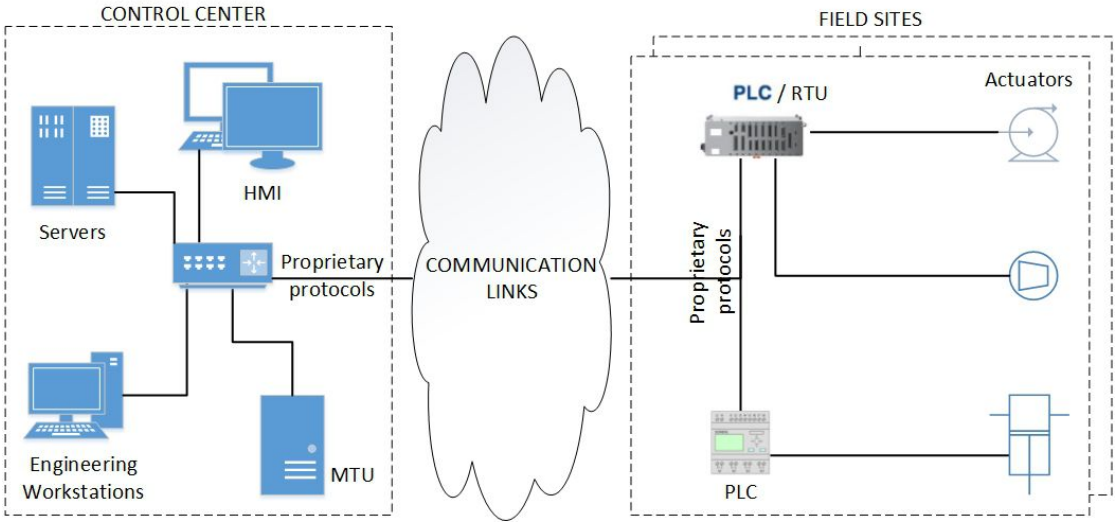

Figure 1.2 – Traditional SCADA Network Architecture

As said in the introduction, the designers of early SCADA systems (Figure 1.2)) were more focused on availability, reliability, performance, longevity [91], than on security. Moreover, the modern architecture of those networks (Figure 1.3) which are linked to corporate networks and to the Internet, as well as encompassing VPNs, COTS hardware, software and standard protocols, exposes them to cyber-attacks like traditional IT ones.

This situation is corroborated by [169], for whom the vulnerabilities of SCADA systems are due to unsafe network architecture, operating system vulnerabilities, backdoor access to unauthorized users, Wi-Fi hardware, lack of real-time monitoring and incorrect encryption, etc. A cyber-attack on SCADA network can take routes from Internet connections, corporate networks, and control

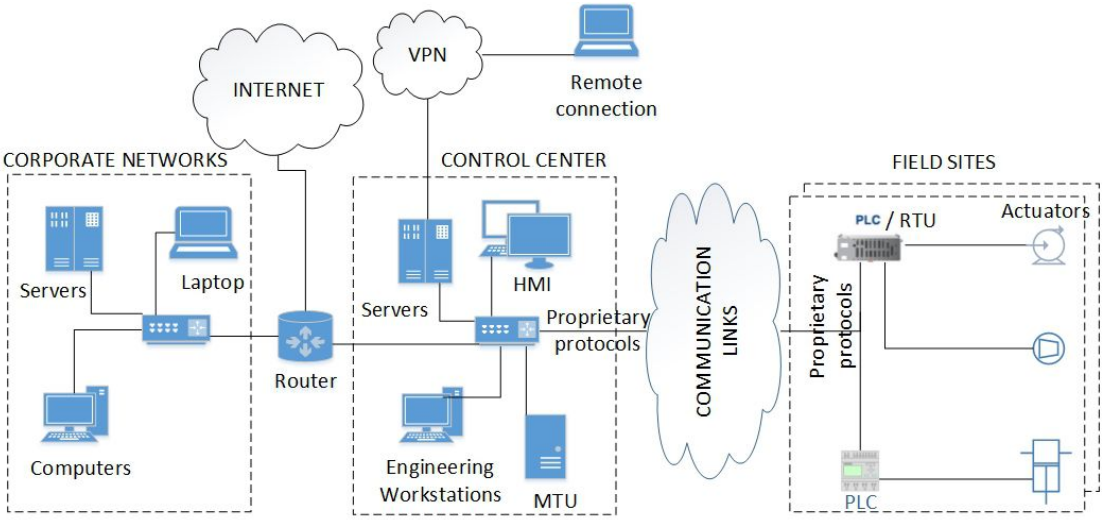

Figure 1.3 – Modern SCADA Network Architecture

networks down to field devices. He also listed various attack vectors such as backdoors and holes in network perimeter, vulnerabilities in common protocols, attack on field devices, database attacks, Man-In-The-Middle attacks, false input data, forged output data, controller historian and Denial-of-Service. [169] [150] make a broad classification of attacks on SCADA networks as follow: attacks on hardware, attacks on software and attacks on the communication stack.

Attacks on hardware

After gaining an unauthenticated remote access to the SCADA devices, the attacker could inject false data like changing the setpoints, causing device to fail or mute an alarm. The attacker can also change the HMI display value to lure the operator.

Attacks on software

SCADA networks software can be targeted by viruses, worms, trojans, botnets. Stack smashing and function pointer manipulation could also cause a buffer overflow and allow an attacker to run his own programs against the SCADA system. Furthermore, the web accessibility of current SCADA systems opens the door to injection attacks, DNS spoofing, session hijacking, phishing, protocol

attacks, application layer attacks, etc.

Attacks on the communication stack

The network layer could be attacked by diagnostic server attacks, idle scan or smurf, the transport layer by attacks like SYN Flood, and the application layer by DNS forgery or command injection attacks [169].

Furthermore, some attacks are specific to the SCADA protocol used. On Gould Modicon Modbus protocol [112] SCADA systems, attackers can perform diagnostic register reset, remote start or slave reconnaissance attacks. Broadcast message spoofing, baseline response replay, direct slave control, network scanning, passive reconnaissance, response delay attacks are also among possible attacks on Modbus protocols. Lastly, TCP-only Modbus SCADA systems could be attacked using irregular TCP framing, TCP FIN Flood or TCP Pool Exhaustion [73]. Other SCADA-specific attacks are related to the DNP3 protocol. Among DNP3-specific attacks, we can mention passive network reconnaissance, baseline response replay, rogue interpoler, length overflow, reset function, unavailable function, destination address alteration, fragmented message interruption, transport sequence modification, outstation write attack, outstation data reset and configuration capture attacks [47].

On the other hand, [53] is grouping SCADA vulnerabilities into four broad categories i.e. the architectural vulnerabilities, the security policy vulnerabilities, the software vulnerabilities and the SCADA communication protocol vulnerabilities. In the architectural vulnerabilities categories, the authors blame the weak separation between the process network and the field network that allow every types of traffic between them. The lack of authentication between actuators-SCADA servers, actuators-RTUs, RTUs-Central production plan system, SCADA -Data exchange servers is also pointed. Poor access policies, traceability and patching are mentioned in the second vulnerabilities category. The third categories of vulnerabilities involves the various operating systems found in SCADA networks such as Linux, SCO Unix, Windows NT, Windows 2000 Server, Windows XP, and

Windows 2003 Server. Those OS as well as the supported applications are not well patched leaving the SCADA servers exposed to attacks. Finally, the last vulnerabilities category i.e. the SCADA communication protocol vulnerabilities are mainly due to TCP/IP version of traditional protocols such as Modbus, DNP3, Profibus and Fieldbus. The shift to modern protocols open new possibilities to attackers because those protocols do not apply integrity checking of commands between Master and slaves, nor do they perform any authentication mechanism between Master and slaves. Anti-repudiation and anti-replay mechanisms are also lacking.

1.4 Documented attacks on SCADA networks

The vulnerabilities detailed in the previous section could be exploited by attackers against the SCADA networks, and many documented attacks are found in the literature [110]. The first known cyber security incident targeting SCADA systems was the Siberian Pipeline explosion in 1982. A Trojan was implanted in a SCADA system that controls the Siberian pipeline. The attack caused an explosion equivalent to 3 kilotons of TNT [38]. Ten years later in 1992, a disgruntled employee of Chevron's emergency alert network disabled the firm's alert system by hacking into computers in New York and San José California. The alarms systems were reconfigured so they would crash. During ten hours, populations in 22 states of the USA and some other areas in Canada were put at risk [42]. With a simple dial-up modem, an attacker gained unauthorized access to the Salt River Project computer network, back in the summer 1994. He then installed a backdoor which enabled him to have access to sensitive information such as water and power monitoring and delivery, financial, customer and personal information [153]. This attack is a perfect illustration of vulnerabilities introduced in SCADA networks by their interconnection with corporate networks. SCADA systems attacks could cause environmental disaster as in June 1999, when a technical incident due to a malfunction of a SCADA gas pipeline controller caused several hundred thousand gallons of gasoline leaked from a pipeline into

a creek in Bellingham, Washington. The gasoline ignited and burned causing 3 deaths and 8 injuries. The loss of human life in this incident illustrates the critical aspect of SCADA systems [152]. This operating anomaly underlines the importance of anomaly detection systems for SCADA networks, which could have detected the incident. A state-sponsored attack was reported by McAfee in February 2011. In fact, they believed that Chinese government backed attackers who conducted a wide attack code-named 'Night Dragon' that targeted five global energy and oil firms over a two years period. This blended attack used a combination of attacks including social engineering, Trojans and Windows-based exploits. The attackers exfiltrated data such as operational blueprints [123]. The most emblematic attack on SCADA systems is named Stuxnet [48]. In June 2010, a worm named Stuxnet attacked a nuclear power plant in Iran. Stuxnet used four 'zero-day' vulnerabilities, insider complicity, stolen certificate. The purpose of Stuxnet was to destroy the centrifuges used by iranian government to enrich uranium. The United States and the Israeli governments are believed to be the perpetrators of the Stuxnet attack. Part of the code of Stuxnet was used the following year in 2011 in the Duqu [11] malware. Duqu did not self-replicate and contained no payload. It was designed to do reconnaissance on an unknown control system. Another sibling of Stuxnet named "Flame" was in action in the Middle East countries and North Africa [11]. It was sponsored by the same group behind Stuxnet. Flame was designed to spy users of infected machines and steal data, including documents, recorded conversations and keystrokes. It also opens a backdoor on the infected system to allow hackers to modify the malware and add new features. On December 23, 2015, the Ukrainian Kyivoblenergo, a regional electricity distribution company was attacked. This attack resulted in a power outage which impacted more than 225.000 customers. The power grid SCADA system were hacked and remotely controlled using various techniques. [102]. More recently in 2018, the world was struck by the Petya and NotPetya malwares [49] which impacted Renault manufacturing plant SCADA systems in France as well as the Germany railway control systems. Those attacks are considered as the most devastating cyber-attacks in history which caused more than 10 billion dollars of financial lost [61].

Many attacks on SCADA systems are perpetrated by insiders or in complicity with them. In 1999, Gazprom, a Russian Oil Company is attacked. The attack was made in collaboration with an insider who was a disgruntled employee. The hackers used a Trojan to take over the system controlling the flow of oil in pipelines. Also, in January 2003, the SQL Slammer worm infected the Davis Besse nuclear power plant in Ohio, USA. This attack resulted in the deactivation for several hours of the system for displaying the safety parameters and the computer for controlling the activity of the control unit [114]. Another insider attack happened in 2000, in Queensland, Australia, where the Maroochy wastewater treatment system, was attacked by a disgruntled contractor because he had failed to get hired in the company [142]. Using a laptop computer and a radio transmitter, the attacker took control of 150 sewage pumping stations and spilled millions of liters of untreated sewage into the environment that caused a major pollution. [100] reports the attack by a night security officer on a heating and ventilation control system at a hospital in Dallas, Texas. He suggests that measures be taken to ensure the physical security of SCADA systems by means of physical access controls and traceability mechanisms for access to control room which will be both deterrent and useful for forensics purpose. The above attacks tends to reinforce the theory of man being the weakest link in computer security. This is proven once again by a team of security experts who used social engineering to deceive employees and take full control of the SCADA network of an electric power company in a matter of hours [62]. The main cause of this attack is the transition of SCADA systems from closed environments to open systems that are interconnected to the Internet and corporate networks. The use of standard software/hardware and the reluctance of companies to apply security measures perpetuate those weaknesses.

1.5 Securing SCADA Systems

Previous sections prove that it is critical to propose reliable solutions to secure SCADA networks. The American National Institute of Standards and Technology (NIST) [145] as well as France ANSSI [108] [24] proposes guidelines for securing SCADA networks. Defense-in-Depth is proposed with network segmentation

and segregation, firewalls; logically separated control network, Unidirectional Gateways, incident detection and response, or system recovery measures. [91] also propose enforcing defense-in-depth by establishing zones and conduits using the ISA 99 i.e. ISA/IEC 62443 standard.

While defense-in-depth is largely encouraged to protect SCADA networks, [58] points out the fact that in industrial sites, priorities are the safety and the reliability of physical processes. The priority of cyber protection of SCADA systems is preventing unauthorized control. Primary, preventive, physical and cyber-perimeter security controls should be the focus. He recommends a Top-Down security approach consisting to deploying unidirectional gateways as sole connection between the protected SCADA system and any outside network. Additionally, removable media and transient devices controls should be implemented. Only after that, secondary controls like detective controls and incident response capabilities can be deployed. Because of the safety/reliability/availability requirements of SCADA systems versus confidentiality/integrity/availability requirements in traditional IT security, deploying security measures in the former ones is not straightforward as in traditional IT. Security solutions should be specific to those environment with regards to their early mentioned specificities. [35] recommend consistent security policy, well-designed network architecture, system hardening by shutting down unnecessary services, ports and software, two-factor password and data encryption for remote connections are also mandatory to secure SCADA systems. Cryptographic methods such as cryptographic algorithms (RC4, DES, AES and RSA) are suggested by [46], Message authentication through Integrity protection with hash and digital signature, Key distribution and Entity authentication are used to achieve confidentiality, integrity, authentication and nonrepudiation security objectives. He also recommend security in communication protocols like PPP, PAP, CHAP, WEP in Link Layer, IPSec in Network Layer, SSL and SSH in Transport Layer, Digest Authentication and PGP in Application Layer. Communications are also secured by the means of Firewalls and Intrusion Detection Systems. Another critical issue in SCADA systems are the communications with business partners such as contractors, vendors and suppliers. As a solution, [125] proposes the authentication of communication partners via hardening passwords or Public Key

Infrastructure (PKI) Technologies. Firewalls are used to protect against external threats whereas an Intrusion Detection System (IDS) can protect against insider as well as external attackers. Internet Protocol Security (IPSec) and Virtual Private Networks (VPN) protect communications between physically distant sites.

1.6 SCADA Intrusion Detection Systems (IDS)

This section is adapted from our Inforsid 2017 conference paper [82].

1.6.1 Signature vs anomaly detection systems

Anomaly detection systems are either signature detection or anomaly detection systems. In the former ones, the system identifies malicious traffic or application data patterns, whereas in the latter ones, the system compares the activities against a "normal" baseline [8] [126]. Each type of detection system has his advantages and drawbacks. The anomaly detection systems are able to detected even new unknown attacks, but are also subject to false alarms (or false positives).

1.6.2 Supervised and unsupervised anomaly detection

Anomaly detection are either supervised or unsupervised. Supervised anomaly detection requires labeled data while in unsupervised approaches there is no labeled data needed. Labels can be extremely difficult if not impossible to obtain, and usually, only a small set of available data can be labeled. Data labeling by human experts is time-consuming, expensive and error-prone. In real applications, we cannot be sure if labeled data cover all possible attacks. Not to say that new attacks could never have been seen in training data [17] [99].

1.6.3 Existing SCADA Intrusion Detection Systems

[120] proposes a Statistical Abnormality Detection Method (SADM), based on the techniques of mean and standard deviation, to thwart internal attacks,

which often have a high impact and high success rate. The system is based on the number of "unresolved alarms" on the operator screen. This number symbolizes the behavior of the operator. [90] developed an approach to detect cyber-physical attacks by applying clustering techniques on industrial process data. The technique used combines a k-means algorithm and subtractive clustering. The architecture of their system uses a Big Data Hadoop infrastructure with the Map/Reduce framework to process large time-series data from several sensors. For the detection of anomaly in process control systems, [154] propose a multilayer and correlation architecture. The proposed system monitors system events at multiple levels (device, network, and hosts) and correlates events at multiple levels (control center, utility and sector levels). The system performs model-based detection, leveraging the regularity and predictability of communication patterns in process control systems. The architecture also uses a hierarchical security incident management framework to correlate IDS alerts and potentially abnormal events generated by the process control system. The solution integrates a visualization tool to help human analysts better understand network traffic anomalies and prevent the defense perimeter from being violated or bypassed. [10] rely on the periodic and regular nature of traffic in SCADA systems to propose an approach to detecting anomalies. The periodicity of the traffic is characterized by the frequency and the size in number of packets of the periodic bursts of the traffic. The system consists of four modules i.e. a traffic analyzer that filters irrelevant traffic, a network flow generation that groups packets meaningfully by using the server-side transport port, a periodicity training module that learns the normal behavior of system by extracting the two elements which characterize the periodic burst, that is to say the period and the size, and finally, a detection engine. To detect attacks injecting false data into control systems, [158] propose an approach based on the relation of states. The proposed approach is a real-time system that monitors system states, detects inconsistent states and deduces the origins of attacks. By means of the relationship graph of the variables, when an abnormal state is detected, one can trace the chain of dependence of violated of the variable (s) involved and deduce the possible origin of the attack. The system architecture consists of three parts which are a component analysis module, a detection module, and an original

inference module. The component analysis module that automatically analyzes system variables to extract components and generates a graph describing the valid system states and another graph of relationships between the variables. Here, alternating vectors which record the alternating relations between two continuous states are also used to represent the real-time states of a component in normal operation. The second module is a detection module that uses the state graph to generate an invalid state alert if a new alternation vector is not found in the state graph or an invalid transition alert if the current state could not be reached from a previous state. Finally, an original inference module helps to locate the origin of the false data injection attacks. The evaluation of the system by Wang et al. gives a detection rate of 95.83% and a false positive rate of (0.0125%). An appliance using a multi-algorithm intrusion detection approach for the Modbus TCP protocol was developed by [30]. This approach is based on a protocol level model, a pattern of expected communication patterns, and a server and service detection model. The protocol level model uses function codes, exception codes, Snort-based policy implementation, and the PVS language to formally specify a specific Modbus device. In the expected pattern of communication patterns, the communication models between the various components of the SCADA network are created, and Snort-based rules are developed to detect deviations from these models. Note that here, the Snort rules are written to detect the "complement" of the models symbolizing normal operation. The last component based on learning to detect changes in server or service availability consists of two detectors that are an Emerald Bayes and EModbus sensor. Experiments show that the model-based intrusion detection approach is effective in monitoring SCADA networks and is complementary to the signature-based approach. Another clustering approach is proposed by [89] for the detection of anomalies in process control systems. This approach is based on the gaussian mixture clustering algorithm. Sensor measurements associated with specific operating states are grouped into clusters, and then the calculation procedure called silhouette is used to detect anomalies. The approach also demonstrates that the algorithm of the gaussian mixture outperforms the k-means clustering algorithm for the detection of anomalies. The proposed system is effective for stealth malicious attacks such as replay, DoS and false data injections.

Table 1.1 – State of the Art of Existing SCADA IDS

Ref.	Research Goal	Detection approach	Detection mode	Outcome/Advantages	Shortcomings
[120]	Detect insider attacks in smart grids	Statistical mean and standard deviation	model-based	Able to detect anomalies in both substation level and transmission system with minimum number of alarms	Requires setting thresholds of normal behavior of the system over time. Some stealthy attacks could go unnoticed as system behavior could fall inside threshold settings
[90]	Use big data technologies for real-time analysis of large dataset to detect cyber attacks in physical systems	k-means and substractive clustering	unsupervised anomaly detection	RMSE = 0.107 Able to process large datasets	The MapReduce slower compared to approaches like Apache Spark for real-time processing. Depends on the ability to conveniently group features of the physical process.
[154]	Leverage regularity of network traffic pattern to achieve anomaly detection in ICS	Use of ruleset and PVS specification language	Model-based and signature-based	Provides timely and accurate reporting of security relevant events	The approach leverages the regularity of network traffic patterns, but some advanced threats could go unnoticed. The signature-based also cannot detect unknown attacks
[10]	Exploit traffic periodicity in ICS in order to perform anomaly detection	Frequency and size of bursts fingerprinting	Model-based	Able to detect frequency anomaly in the time domain	The approach is based on the periodicity of ICS traffic to detect anomalies. Could be defeated by advanced threats
[158]	False data injection attacks detection and retrieval of attack origin	Use of state alternation vectors and state relations graph	Rule-based	Detect. Rate=95.83 % FPR = 0.00125 % Good detection rate and very low FPR	Attacker can modify initial state of the detection system and thus not be detected.
[30]	Take advantage of regularity and stability of ICS to implement model-based IDS	PVS specification language and conditional probability	Model-based	Effective for monitoring SCADA networks	Based on regularity and stability of ICS . Could be defeated by advanced threats
[89]	Detected cyber attacks targeting measurements sent to control hardware (PLCs)	Gaussian mixture clustering and silhouette cluster evaluation technique	Unsupervised anomaly detection	Average silhouette=0.37067	The approach is implemented on PLCs or other remote I/O devices, which have low memory and processing power.

1.7 Conclusion

In the present chapter, we showed that security was not a major concern for the designers of SCADA networks. But as time evolves, SCADA systems which were using vendor-specific hardware, software and protocols, incorporated in their architecture Commercial-Off-The-Shelf (COTS) hardware, software, as well as standard protocols. Moreover, modern SCADA systems are interconnected to corporate networks and to the Internet. This shift in SCADA networks introduced various vulnerabilities that we explained, and exposed diverse documented attacks which exploited those vulnerabilities. As countermeasures to those attacks, we provided methods to secure SCADA systems and we made a focus on SCADA specific anomaly detection systems.

Due to the evolving threat landscape, with more and more new complex attacks like blended attacks [91] and Advanced Persistent Threats (APT), we need more advanced approaches that could be more reactive and able to adapt to the new deal. Deep learning which has the capability to automatically learn the intrinsic characteristics from a system data could help build anomaly detection systems to protect SCADA networks from unknown attacks.

In the next chapter, we provide the reader some insights of deep learning fundamentals, common architectures, and their unsupervised feature learning capability that could be used for anomaly detection purpose.

Chapter 2

Deep Learning Overview

2.1 Introduction

In 1959, Arthur Samuel [139] defines Machine Learning as a field of study that gives computers the ability to learn without being explicitly programmed. Later on in 1997, Tom Mitchell [111] gave a more formal definition of Machine Learning by stating that *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"*.

Deep Learning is a sub-field of Machine Learning which is using deep neural networks i.e. neural networks with at least two hidden layers [14] [57]
In recent years, Deep Learning became a hot topic among researchers as this approach is successfully applied to domains like image classification, video and natural language processing.

The objective of this chapter is to provide the reader the foundations of deep learning i.e. weights, bias, number of layers, loss and activation functions, hyper-parameters or back-propagation algorithm, etc. After those knowledge which are important to understand the design principle of deep learning networks, we will show the most common deep learning architectures and how they could be used for feature learning prior to classification tasks.

The first section is dedicated to explaining the different concepts of artificial neural networks, followed by the study of the process of training deep networks mainly using the back-propagation algorithm with Gradient Descent. Thereafter, we explore the main deep learning networks architectures before the conclusion.

2.2 Artificial Neural Networks

The artificial neural networks mimics the human brain that is processing information in a totally different way from the conventional digital computer. The brain is a complex, nonlinear, and parallel computer which has the capability to organize its basic components known as neurons, so as to perform computations many times faster than today's best computers [64].

2.2.1 Biological Neuron

The biological neuron is a nerve cell that provides the fundamental functional unit for the nervous systems of all animals. Neurons communicate with each other by the mean of electro-chemical impulses. The impulse must be strong enough i.e above a minimum threshold to activate the release of chemicals. The neuron cell body called soma has many dendrites but only one axon that is able to branch hundreds of times. Dendrites are thin structures that arise from the main cell body, while axons are nerve fibers with a special cellular extension that comes from the cell body [128].

2.2.2 Single Layer Perceptron

The perceptron is the first attempt to model the biological neuron. The perceptron algorithm was invented in 1958 at the Cornell Aeronautical Laboratory by Frank Rosenblatt [135]. The perceptron is a linear-model binary classifier with a simple input–output relationship. A number of inputs are multiplied by their weights and the result is sent to a step function (Heaviside). The classification can be modeled as follow [128] :

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x >= 0 \end{cases}$$

2.3 Deep learning foundations

Now, let us give some explanations about the fundamental concepts of deep learning i.e. weights and bias parameters, the input, hidden and output layers, the activation and lost functions, as well as the various Hyper-parameters needed to tune deep networks [128] [121] [141].

2.3.1 Parameters

The parameters of neural network are *weights* and *bias*. *Weights* are real valued numbers which are multiplied by the inputs and then summed up in the node. *Bias* terms are constants attached to neurons and added to the weighted input before the activation function is applied. The basic idea is that is it possible there is a threshold upon which your features have an effect. This value is the bias which is coming along with the weights and must also be learned during the model training process.

2.3.2 Input, hidden and output layers

The *Input layer* represent how we get input data fed into the network. It has the same number of neurons as the number of input features. Each neuron in the input layer represents a unique attribute in your dataset. On the other hand, There is one or more *hidden layers* in a neural network. The weight values on the connections between the layers are how neural networks encode the learned information extracted from the raw training data. Hidden layers introduce non-linearity modeling in neural networks. The *Output layer* is the final layer in a network. It receives input from the previous hidden layer, applies an activation function, and returns an output representing your model's prediction. The answer or prediction from the model is in the output layer. The final output may be a real valued output (regression) or a set of probabilities (classification).

The output depends on the type of activation function we use on the neurons in the output layer, which is usually a softmax or sigmoid activation function for classification.

2.3.3 Activation Functions

The role of activation functions (Table 2.1) is to enable neuron's activation, in other words, help the neurons to fire. They play a primary role in hidden layers as they introduce non-linearity, thus increasing neural networks modeling capabilities [3]. There are different types of activation functions, and their use depend of the problem to solve.

The *Linear* activation function is actually the identity function expressed by $f(x) = Wx$. The Linear activation function let the signal passed unchanged through the neuron. The dependent variable has a direct, proportional relationship with the independent variables. This type of activation function is often used in input layer of neural networks [141]. However, linear activation functions has two major drawbacks. First, it is not possible to use back-propagation (gradient descent) to train the model as the derivative of the function is constant. Second, all layers of the neural network collapse into one no matter their number, and the output is linear because a combination of linear functions is linear [1] [128].

The *Sigmoid* activation function is a logistic transforms type function. Sigmoid can reduce extreme or outliers in data without removing them. Independent variables are converted to simple probabilities between 0 and 1, and most of the output will be very close to 0 or 1. Advantages of sigmoid activation function is its smooth gradient, the output bound between 0 and 1, which normalize the output of each neuron, and finally its clear predictions. The sigmoid drawback is the vanishing gradient problem, the output not centered zero and its computationally expensive [1] [128] [141].

Unlike the sigmoid activation function, the normalized range of the *Tanh* activa-

tion function is -1 to +1. The advantages of Tanh is that it can deal more easily with negatives numbers and is zero centered unlike sigmoid. They share the same disadvantages. A variant of Tanh is *Hard Tanh* where anything less that -1 is set to -1 and anything more than1 is set to 1. Hard Tanh activation function is a more robust activation function that allows for a limited decision boundary [128] [1] [83].

Softmax activation function is a generalization of logistic regression that can contain multiple decision boundaries. Often used in the output layer of classifiers, the softmax activation function return the probability distribution over mutually exclusive output classes. The returned probabilities sum-up to 1. The main advantage of softmax is its ability to handle multiple classes. [1] [128].

The *Rectified Linear Unit (ReLU)* which equation is $f(x) = max(0, x)$ activates node only if the input is above a certain quantity. While input is below zero, output is zero. When input is above a certain threshold, it has a linear relationship with the dependent variable. ReLU is the current state of the art. It is proven to work in many different situations. Gradient of ReLU is either 0 or constant, making it possible to reign with the vanishing exploding gradient issue. ReLU activation functions have shown to train better in practice than sigmoid activation functions. Compared to the sigmoid and tanh activation functions, the ReLU activation function does not suffer from vanishing gradient issues. Relu is computationally efficient and allow the network to converge quickly [140] [37]. ReLU is also non-linear though looking as a linear function. It has a derivative and thus allow backpropagation. However, ReLU suffers from the Dying ReLU problem. When inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn [1].Various forms of ReLU such as Leaky ReaLU (LReLU) [106], Parametric ReLU (PReLU) [66] help overcome the aforementioned problem.

The summary of activation functions in Table 2.1 indicates that to design an efficient deep neural network, one should consider using the ReLU or Leaky ReLU activation function in the hidden layers instead of sigmoid or Tanh, be-

Function	Mathematical Model	Range	Advantages	Limitations
Linear	$f(x) = ax$	]-inf, +inf[	- Not complex - Easy to solve	- Have less learning power - Does not perform good most of the time
Sigmoid	$\frac{1}{1+e^{-x}}$	[0, 1]	- S-shape curve - Easy to understand and apply - Output as probability	- Vanishing Gradient Problem - Not zero-centered making optimization harder - Saturate and kill gradient - Slow convergence
Tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	[-1, +1]	- Output zero-centered making optimization easier - Prefered to sigmaoid	- Vanishing Gradient Problem
ReLU	$f(x) = max(0, x)$	[0, +inf[	- Very simple - Computationally efficient than sigmoid and Tanh - Avoid Vanishing Gradient Problem	- Output not zero-centered leading to slow convergence - Only used on hidden layers - Dying ReLU problem
Leaky ReLU	$f(x) = \begin{cases} 0.01x & x < 0 \\ x & x >= 0 \end{cases}$	]-inf, +inf[	- All advantages of ReLU - Fix Dying ReLU Problem - Output zero-centered, thus converge faster	Slope parameter added
Softmax	$f(x_j) = \frac{e^{x_j}}{\sum_i e^{x^i}}$	[0, 1]	- Multiclass classification - maps logits to probabilities	- Only used on output layers

Table 2.1 – Activation Functions

Loss Function	**Type of Problem**	**Scalar/ Binary/ Multiclass**
MAE/L1 Loss	Regression	Scalar
MSE / L2 Loss	Regression	Scalar
Hinge Loss	Classification	Binary
Maximum Likelihood	Classification	Multiclass
Cross Entropy	Classification	Multiclass
Kullback-Leibler	Reconstruction	NA

Table 2.2 – Loss Functions

cause of the vanishing Gradient Problem which causes a lots of problems to train, degrades the accuracy and performance. Furthermore, Leaky ReLU converges faster, which could help reduce the training complexity challenge. For classification problems as in anomaly detection systems, the Softmax activation function should be used in the output layer.

2.3.4 Loss Functions

A loss function is a metric based on the error of a network's predictions. Training a neural network consists in finding the ideal parameters (weights and biases) that minimize the loss from the errors. With loss function, training a neural network is solving an optimization problem. The loss function to use when training a neural network depends of the class of problem we're trying to solve i.e. regression, classification or reconstruction (Table 2.2).

Mean Square Error (MSE), Quadratic Loss, L2 loss, Mean Absolute Error (MAE), L1 loss [107] [75] [79] [34] are loss functions- for regression problems where a scalar value is the outcome.

However in intrusion/anomaly detection approaches, the problem to solve is a classification one as the system might be able to classify normal data from anomalous ones.
For classification problems Hinge Loss or Cross Entropy Loss also called Nega-

tive Loss Likelihood [107] [75] [79] [34] are used. However, while Hinge Loss is most suited for binary classification [79], Cross Entropy Loss is the most widely used loss function for multiclass classification.

Cross Entropy Loss

To measure the information theoretic distance between two distributions $P = \{p_1, p_2, ..., p_N\}$ and $Q = \{q_1, q_2, ..., q_N\}$, Kullback [95] proposes the directed divergence now know as cross-entropy, formulated by:

$$D(P,Q) = \sum_{k=1}^{N} q_k log_2 \frac{q_k}{p_k}$$

In machine learning, the *Cross entropy* allows the model to estimate the conditional probability of the classes, given the input, and pick the classes that minimizes classification error. Cross-entropy loss increases as the predicted probability diverges from the ground truth label. [59] [121] [79].

Kullback-Leibler Divergence (KL Divergence)
Deep learning-based anomaly detection systems also use Kullback-Leibler Divergence [96] to introduce sparsity parameter. The KL divergence measures the divergence between two Bernoulli distributions.

$$D_{KL}(Y\|\hat{Y}) = -\sum_{i=1}^{N} Y_i \times \log\left(\frac{Y_i}{\hat{Y}_i}\right)$$

2.3.5 Hyper-parameters

Hyper-parameters are parameters we tune to make networks train better and faster. There is two kinds of Hyper-parameters : the ones related to the neural network structure and those related to the training algorithm [76]. The Hyper-parameters related to neural network structure are the *Number of hidden layers*, *Dropout*, *Neural network activation function*, and *Weights initialization* For the *Number of hidden layers* hyper-parameter, usually, adding more hidden layers of neurons generally improves accuracy to a certain limit. *Dropout* is a technique that consists at randomly "killing" a certain percentage of neurons during each epoch to prevent overfitting.The *Neural network activation function* is also considered as an hyper-parameter. The choice of the activation function depend of the class of problem we are solving. The activation function can impact the network's ability to converge and learn for different ranges of input values, and also its training speed. *Weights initialization* is a very important factor while training neural networks. At the beginning of the training process of a neural network, weights and bias should be initialize. Weights can be set to zero or to random values. However, this can result in a vanishing or exploding gradient, which will make it difficult to train the model. A common heuristic used to mitigate this problem for the Tanh activation is called *Xavier initialization*.
The other type of hyper-parameters is *Hyper-parameters related to the training algorithm* which encompasses *Learning rate*, *Epoch*, *iterations and batch size*, *Optimizer algorithm and neural network momentum*. The *learning rate* determine how fast the back-propagation algorithm performs gradient descent. A lower learning rate makes the network train faster but might result in missing the minimum of the loss function.After setting the learning rate, *Epoch*, *iterations*

and batch size have to be defined. These parameters determine the rate at which samples are fed to the model for training. An epoch is a group of samples which are passed through the model together (forward pass) and then run through back-propagation (backward pass) to determine their optimal weights. The epoch can be split into batches before being fed to the network in order to reduce complexity or in case of big data size. A complete epoch will then be run into multiple batches. The algorithm used to train a neural network is called an *optimizer*. The basic optimizer is *Stochastic Gradient Descent*. The *Momentum* algorithm is also a popular choice. It works by waiting after a weight is updated, and updating it a second time using a delta amount. This speeds up training gradually, with a reduced risk of oscillation. Nesterov Accelerated Gradient, AdaDelta and Adam also can be used.

2.3.6 Regularization

The regularization main purpose is to prevent *overfitting* during the training process by using different methods to minimize parameter size over time. Some of the techniques used to this end is L1, L2 (weight decay), dropout or Early stopping [75] [59].
The regularization term is added in order to prevent the coefficients to fit so perfectly to overfit. The *L1 regularization* is a technique that help to penalize the size of the weights by adding a regularization term

$$\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$$

to the objective function

$$w^* = arg\ \min_w \sum_j \Big(\ t(x_j) - \textstyle\sum_i w_i h_i(x_j) \ \Big)^2 + \lambda \displaystyle\sum_{i=1}^{k} |w_i|$$

In *L2 regularization*, L2 parameter norm penalty commonly known as *weight decay*. This regularization strategy drives the weights closer to the origin by adding a regularization term $\Omega(\theta) = \|w\|_2^2$ to the objective function

$$w^* = arg\ \min_w \sum_j \Big(\ t(x_j) - \textstyle\sum_i w_i h_i(x_j) \ \Big)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

The L2 regularization is more computationally efficient compared to L1 regularization due to having analytical solutions.
Dropout provides a computationally inexpensive but powerful method of regularizing a broad family of models. Dropout trains the ensemble consisting of all sub-networks that can be formed by removing non-output units from an underlying base network [59]. Finally, *Early stopping* Early stopping is thought to be the most commonly used form of Deep Learning regularization. When training large models we often observe that training error decreases steadily over time, but validation set error begins to rise again. The main idea is to return to the parameter setting at the point in time with the lowest validation set error. Instead of running the optimization algorithm until we reach a (local) minimum of validation error, we run it until the error on the validation set has not improved for some amount of time. A copy of the model parameters is store every time the error of the validation set improves. When the training process is finished, the stored parameters are return in lieu of the latest parameters [59].

2.4 Training Neural Networks

2.4.1 Gradient Descent Optimization

Most Deep Learning algorithms involve optimization of some sort. Optimization refers to the task of minimizing a function called the objective function, criterion, cost function, loss function or error function [59]. Gradient Descent is by far the most common way to optimize neural networks. Let's the objective function be $J(\theta)$ with $\theta \in Rd$ the parameters of the model. Gradient descent is a way to minimize $J(\theta)$ by updating the parameters in the opposite direction of the gradient of the objective function $\Delta_\theta J(\theta)$ with respect to the parameters [137]. In gradient descent also known as batch gradient descent, we would calculate the overall loss across all of the training examples before calculating the gradient and updating the parameter vector. Other variants exist such as Stochastic

Gradient Descent (SGD) or mini-batch gradient descent. In SGD, we compute the gradient and parameter vector update after every training sample. SGD speed-up learning and also parallelizes well. In between Batch gradient descent and SGD lies mini-batch gradient descent, where a batch of samples more than a single training example and less than the full training dataset is used to compute the gradient.
The loss function can be minimized by estimating the impact of small variations of the parameter value on the loss function.
Let's W be the set of parameters and $E(W)$ the loss function. The gradient descent algorithm allow the minimization of $E(W)$ by iteratively adjusting W as follow [101].

$$W_k = W_{k-1} - \epsilon \frac{\partial E(W)}{\partial W}$$

Where ϵ is a scalar constant called the learning rate. This is the *Weights update with Gradient Descent*.

2.4.2 Back-propagation Algorithm

In a well-trained ANN, the weights amplify the signal and dampen the noise. A bigger weight means a tighter correlation between a signal and the network's outcome. The process of learning using weights is the process of re-adjusting the weights and biases. Back-propagation learning computes the input example's output with a forward pass through the network. If the output matches the label, we don't do anything. If the output does not match the label, we need to adjust the weights on the connections in the neural network [127]. The key is to distribute the blame for the error and divide it between the contributing weights in a backward pass.

The back-propagation algorithm pseudo-code is given bellow by [128] where i is the index of neuron, n_i the neuron at index i, j the index in previous layer connecting to j, a_i the activation value of neuron i (output of neuron i), A_i the vector of activation values for the inputs into neuron i, g the activation function g' the derivative of the activation function, Err_i the difference between the network output and the actual output value for the training example, W_i the vector of

weights leading into neuron i, $W_{j,i}$ the weights on the incoming connection from the previous layer neuron j to neuron i, $input\_sum_i$ the weighted sum of inputs to neuron i, $input\_sum_j$ the weighted sum of inputs to neuron j in previous layer, a the learning rate, Δ_j the Error term for connected neuron j in previous layer and $\Delta_i$the error term for neuron i: $\Delta_i = Err_i \times g'(input\_sum_i)$.

Algorithme 1 : Back_propagation_algorithm

Data : network, training_records, learning_rate
Result : network
$network \leftarrow initialize\_weights(randomly)$
while *network not converged* **do**
  foreach $example \in training\_records$ **do**
    /* compute the output for this input example */
    $network\_output \leftarrow neural\_network\_output(network, example)$
      /* compute the error and the delta for neurons in the output layer */
    $example\_err \leftarrow target\_output - network\_output$ /* update the weights leading to the output layer */
    $W_{j,i} \leftarrow W_{j,i} - \alpha * a_j * Err_i * g'(input\_sum_i)$
    foreach *subsequent layer* $\in$ *network* **do**
      /* compute the error at each node */
      $\Delta_j \leftarrow g'(input\_sum_j) \sum_i W_{j,i} \Delta_i$
      /* update the weights leading into the layer */
      $W_{k,j} \leftarrow W_{k,j} + \alpha * a_k * \Delta_j$
    end
  end
end

2.5 Deep Learning architectures

2.5.1 Feed-forward neural network

A Feed-forward neural network is a neural network ordered in layers, where the first layer is called input layer, the last layer output layer, and the layers between are hidden layers [146]. Connections between the nodes do not form a cycle. An example of feed-forward neural network is the Multi-Layer Perceptron (MLP) [136].

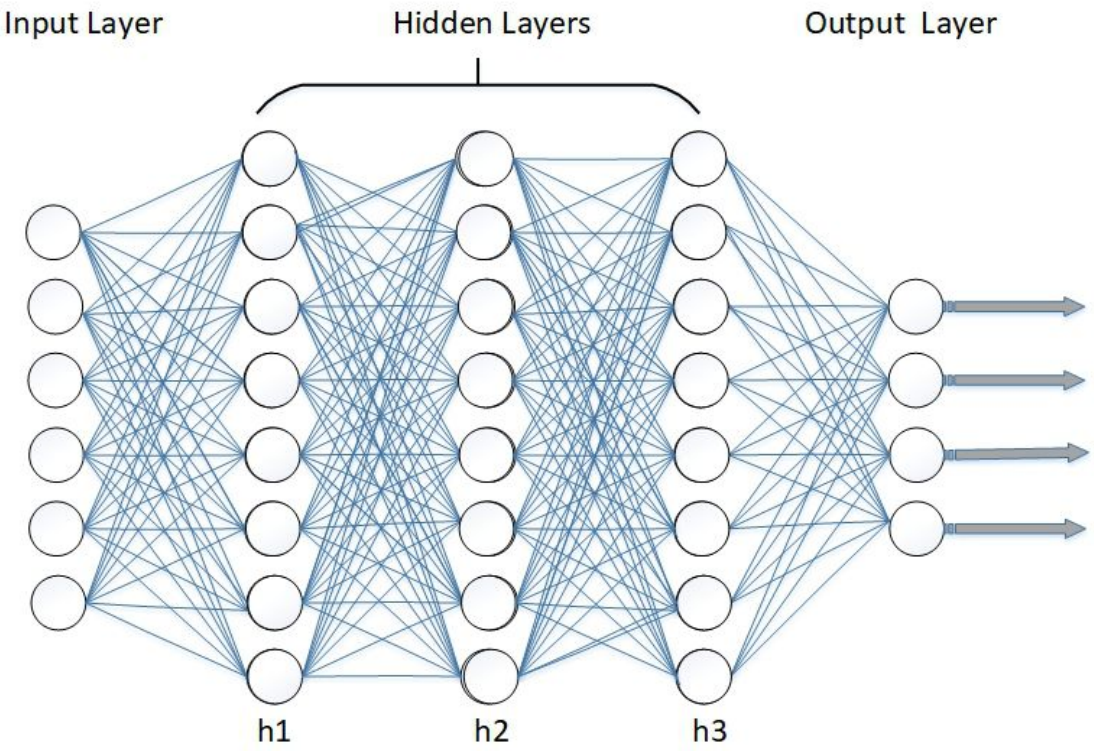

Figure 2.1 – Feed Forward Neural Network

2.5.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) goal is to learn higher-order features in the data via convolutions. The convolution operation is used instead of general matrix multiplication in at least one of their layers [59]. The convolution operation (Figure 2.2) is a mathematical operation on two functions of a real-valued

argument. In CNN, the first argument is called input, the second argument the kernel and the output feature map. For a two-dimensional image I and a two-dimensional kernel K, the convolution operations is

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

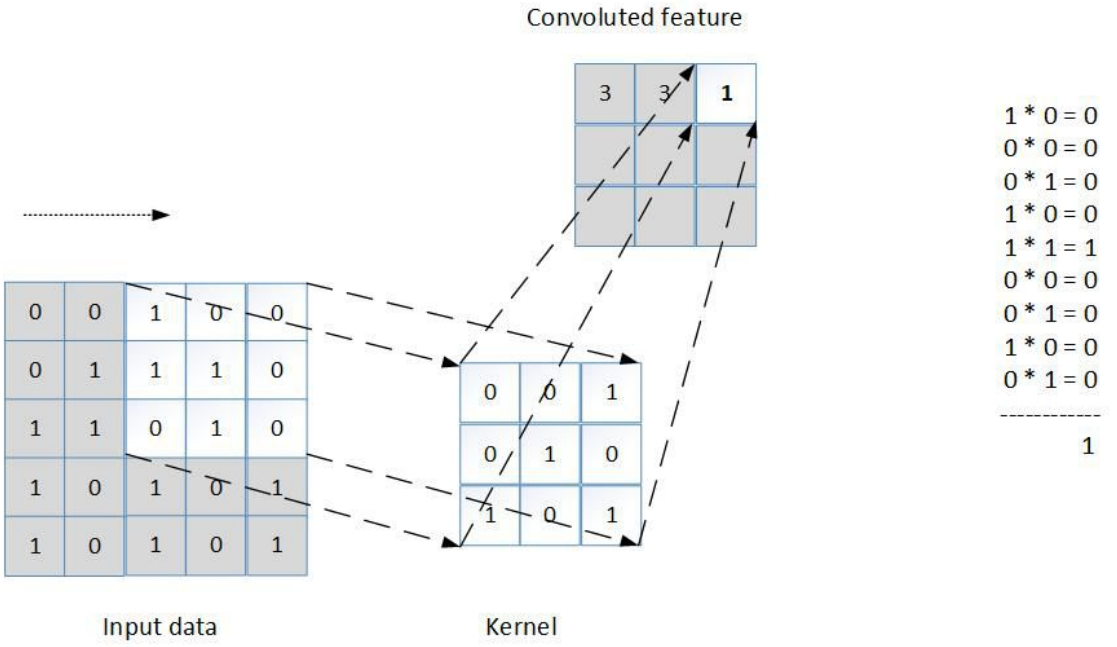

Figure 2.2 – Convolution Operation

CNNs work well for object recognition with images. CNN architecture biological foundation is the visual cortex in animals. The cells in the visual cortex are sensitive to small sub-regions of the input called visual field or receptive field. These smaller sub-regions are tiled together to cover the entire visual field. The typical components of a CNN are an input layer, a convolution layer, an activation function (generally ReLU), a pooling layer and a fully connected layer [127] [128] [93]. The combination of convolution layer, activation function and pooling layers is the feature extraction layer. Two important characteristics of CNNs are weight sharing and pooling. Weight sharing is the fact that all the spatial locations of the input share the same kernel, whereas pooling refers to an operation reducing the number of connections between convolutional layers,

thus decreasing the computational burden [113]. The *Fully connected layers* on the other hand contain traditional neurons that receive different sets of weights from the preceding layers; there is no weight sharing between them as is typical for convolution operations.

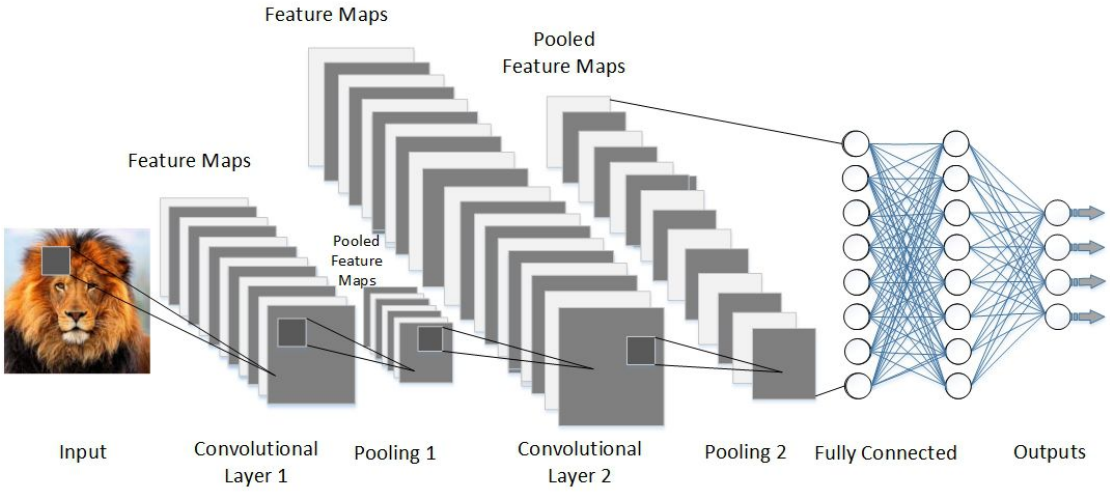

Figure 2.3 – Convolution Neural Network

2.5.3 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) are dynamic systems with internal state at each time step of the classification. This is because there are circular connections between higher- and lower-layer neurons and self-feedback connections. Thanks to the feedback connections, RNNs are able to propagate data from earlier events to current processing steps. Thus, RNNs build a memory of time series events. However, because of Vanishing Gradient Problem, RNNs are not capable to learn distant dependencies. Hence, RNNs cannot bridge 5 to 10 time steps. This is why alternatives like LSTM are proposed to solve this issue. LSTM can bridge minimal time lags of more than 1,000 discrete time steps [144]. RNNs are supervised approaches but are used as unsupervised methods in some applications. They are suited to model time-series data or Natural Language Processing (NLP).

2.5.4 Generative Adversarial Networks (GAN)

Goodfellow and his team [60] proposed a new framework consisting in training two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The goal of the model G training is to maximize the probability of D making a mistake. GANs are mainly used in creating realistic images, paintings, speech processing, and video clips.

2.5.5 Deep Neural Networks (DNN)

Deep Neural Networks are hybrid architecture combining both generative and discriminative models. The final goal of DNNs architecture is to discriminate data, however, in previous stages of the process, it is assisted by features learned from generative models[87].
There are different architectures of deep neural networks whose building blocks are feed-forward neural network, Restricted Boltmann-Machines (RBM) or Autoencoders (AE) [119] [70]. In a DNN model, RBMs or Auto-encoders are stacked for feature learning, and a supervised classifier such as a Multilayer Percepron (MLP), Random Forest, Support Vecor Machine (SVM) or a Softmax layer is added on the top for classification.

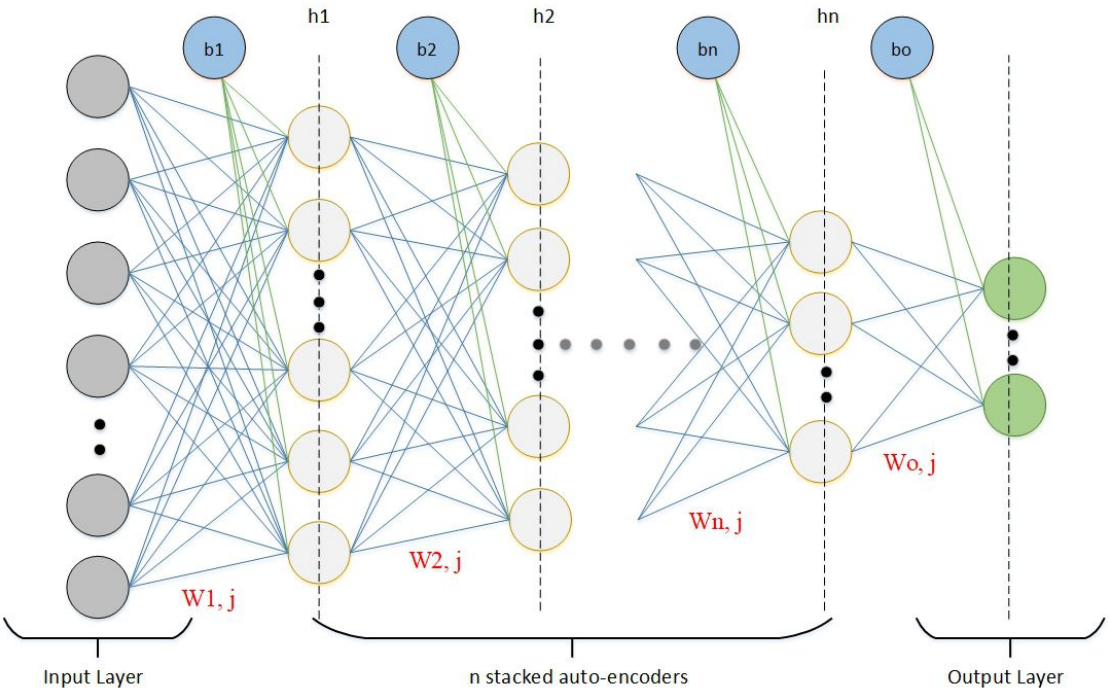

Figure 2.4 – Stacked Auto-encoder

2.6 Conclusion

In this chapter, we have learned the foundations of deep learning networks, i.e. their parameters (weights, bias), loss and cost functions, the different of layers, and their hyper-parameters. We found that care should be given to activation and loss functions choices as they depend of the nature of the problem to solve. Some techniques should also be applied to avoid overfitting. The training process of deep learning networks turns to be an optimization problem as it consists in minimizing the loss function. The backpropagation algorithm which is using the Gradient Descent method is widely used to train deep networks. Finally, we covered various deep learning architectures such as CNN, RNN, GAN and DNN.

Even though training deep learning networks is quite challenging in terms of processing time, their automatic feature learning capabilities [59] [87] [4] make them clearly interesting for data-driven anomaly detection for SCADA networks, as labeling the huge data generated by such networks is hardly feasible.

To find out how deep learning approaches leveraging automatic feature learning techniques is used in SCADA networks, we will conduct in the next chapter a review of existing SCADA anomaly detection systems using deep feature learning approach.

Part II

Contributions

Chapter 3

Review of SCADA Anomaly Detection Systems using Deep Feature Learning Approach

3.1 Introduction

The first chapter of this thesis highlighted the vulnerabilities of SCADA networks that could be exploited by attackers. Intrusion and anomaly detection systems play a key role among other security systems.
The present chapter is a review of deep learning feature-learning based approaches used for anomaly detection in SCADA systems. Its goal is to show that the automatic feature learning capability of deep learning approach can be used to provide anomaly detection in SCADA networks. We have included two studies related to anomaly detection caused by operating faults in this review because an operational anomaly could be a consequence of an attack as it was the case when in 2010 the Stuxnet attack crippled the Iranian uranium enrichment centrifuges [48].

In the first section, we make a presentation of feature learning technique, followed by a review of deep learning-based Intrusion detection systems in SCADA networks using feature learning. For each topic, we present the problem domain,

the deep learning approach used, the dataset and the experimentation results. Finally, we make a summary of the different approaches before concluding.

3.2 Unsupervised Feature Learning

In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon being observed [16]. In standard machine learning, feature learning from data is a complex task as it requires experts of the domain to handcraft the original features in order to feed the machine learning algorithms with the best features. The data learning process could be supervised or unsupervised. The supervised learning also need the intervention of human to correctly label the data, which is costly and error prone.
In order to leverage the huge amount of available unlabeled data, deep learning algorithms can automatically learn important features from data in an unsupervised manner [59] [162] [87] [4]. . The main purpose of unsupervised feature learning is to provide a function to map the original set of features into a different representation [165].

In real world, most data are complex, and building good predictors on those data means learning complex functions too, which are best represented by multiple levels of non-linear operations, i.e. deep architectures [155]. Unsupervised feature learning can be done by using clustering on data using algorithms such as K-means [13], or by training stacked auto-encoders or convolutional networks [156].

3.3 Review of unsupervised feature learning in SCADA ADS

3.3.1 LSTM/Bloom filter anomaly detector

In order to detected anomalies due to data/command injection, reconnaissance or Denial-of-Service (DoS) attacks on a gas pipeline SCADA system, [50] propose an anomaly detection approach consisting of a Bloom filter anomaly detector and a time-series level anomaly detector (Figure 3.2). The former one is a packet-level anomaly detector which checks a packet signature in its database. The database stores network patterns and communication pattern signature as they are stable in a SCADA system. If the analyzed package signature is not in the Bloom filter, the packet is considered anomalous. The latter detector receives normal packet that pass the Bloom filter for a time-series level anomaly detection, which uses its power of information memorization for an extended number of time steps to predict the behavior of the next time step. a Bloom filter is a probabilistic data structure that is used to test whether an element is a member of a set. It is commonly used as an in memory data structure which size is limited by the availability of RAM space on the machine [40]. Because of the limited memory and computing resources of some of SCADA components, using a Bloom filter as a fast and light-weighted packet level anomaly detector is important. It efficiently stores the signature database of normal network packets and detects anomalies thereafter. On the other hand, the time-series anomaly detector is a Stacked Long Short Term Memory (LSTM) Network-based Anomaly Detector (Figure 3.1) which takes the input of time-series $x(t-1), x(t-2), ...$, learns their higher dimensional feature representations, and then uses those features to predict the next data point $\hat{x}(t)$. Furthermore, the predicted data point can be used to classify if $x(t)$ is anomalous by checking the similarity between $x(t)$ and $\hat{x}(t)$. The LSTM network model is then trained to minimize a softmax loss function suited for multi-class classification [59] [128].
The evaluation of the combined anomaly detection framework on a gas pipeline SCADA dataset [115] gives an accuracy of 92 %, which is higher compared to other approaches. However, the training time of the LSTM model of 35 min

during 50 epochs is rather high.

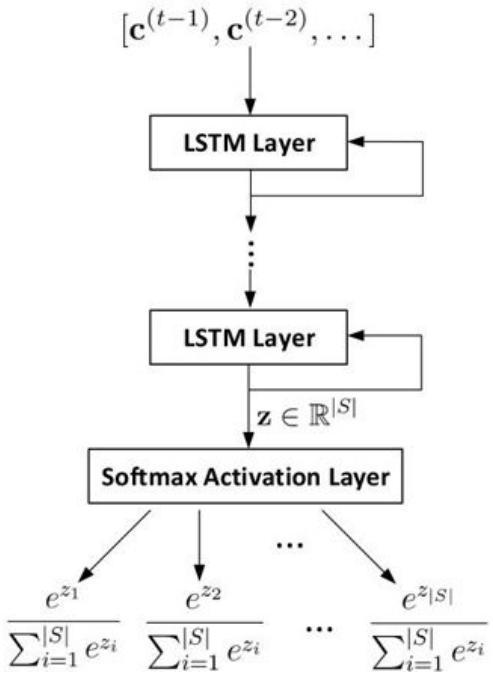

Figure 3.1 – Architecture of the stacked LSTM-based softmax classifier model
Source: [50] ©2017 IEEE

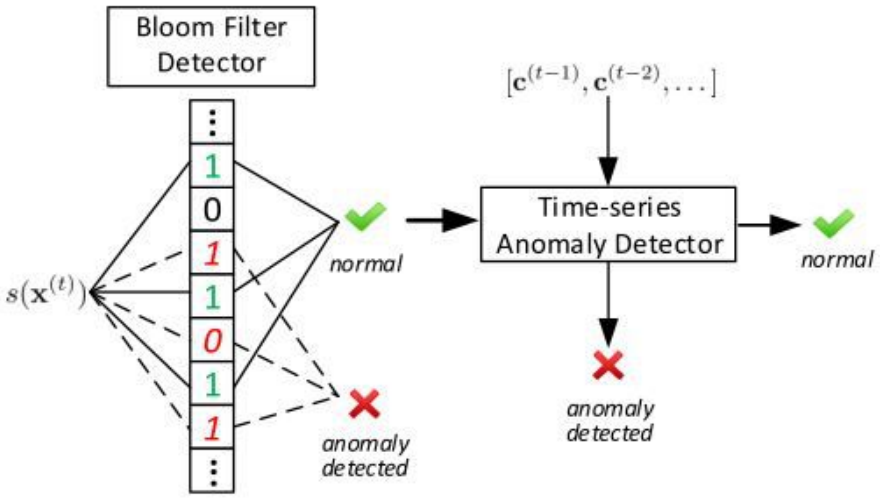

Figure 3.2 – Combined framework for package and time-series level anomaly detection
Source: [50] ©2017 IEEE

3.3.2 Stacked Auto-encoder based anomaly detection

Due to Network bandwidth and network data increase, [132] proposes a deep packet inspection in order to extract the necessary feature that would allow DoS, Probe, R2L and U2R attacks detection. The authors used a Deep Neural Networks (DNN) approach which architecture is a stacked auto-encoders for the feature learning, to which a softmax layer is added for the classification (Figure **??**). The stacked auto-encoder has two hidden layers, one with 20 nodes and the second with 10 nodes. The dimension of the learnt features is 10 compared to the 41 original features of the NSL-KDD dataset dataset. The overall process encompasses four steps i.e. a feature learning step with the stacked auto-encoder, a first fine-tuning step where the softmax layer is trained in a supervised manner with labels and training data. The input of this first fine-tuning step is the compressed representation of the data. The following step is a second fine-tuning with a back-propagation training applied to the whole network layers after the first fine-tuning step. The goal of this second fine-tuning step is to refine the features of the intermediate layers to make them more relevant for the intrusion detection task by adjusting the network weights to minimize the cost function. Finally, the last step of the process is a classification and testing step where a test dataset is presented to the fine-tuned network in order to evaluate the efficiency of the model. Accuracy, precision, recall and f-mesure metrics are used to evaluate the proposed approach against standard techniques like k-means, DBN, SOM, AdaBoost.
Experimental results show that despite good detection accuracy for DoS and Probe attacks (97.6 % and 86.34 % respectively), R2L and U2R attacks give poor results (12.98 % and 39.62 % respectively). The poor performance of the latter two categories of attacks is due to the lack of sufficient amount of data related to R2L and U2R (0.04 % and 0.79 % respectively). 9 to 10 % training data samples for R2L and U2R categories of attacks as with the probe attacks would have given better detection results. However, the approach proposed by [132] gives promising results in feature learning and good detection rate for some classes of attacks detection. It uses the NSL-KDD dataset, an improved version of KDD Cup 99 [149] which is a general bench-marking dataset for network intrusion

detection research, created twenty years ago. Those datasets may not reflect modern networks traffic complexity nor integrate new complex attacks.

3.3.3 Stacked Auto-encoder for anomaly detection in smart grids

The cyber-physical integration, exposes smart grids which are critical systems to a ubiquitous attack surface through which exploits may inflict major disruptions or damages. There is a high demand for advanced situational awareness (SA) to provide early warnings and protect electric utilities against adversaries from the cyberspace. Among the countermeasures against such attacks, Intrusion/Anomaly Detection Systems play a key role [8]. Machine learning approaches are used to develop data-driven anomaly detection systems. However, human handcrafted features for machine learning anomaly detectors become more expensive and less effective in smart grid [103] [124]. This situation led [161] to use a stacked auto-encoder approach to supplement more high-quality feature for ML-based threat monitoring (Figure 3.3). The approach has two main phases: An off-line training phase and an online monitoring phase. During the off-line training phase, historical data are first collected for training purpose on different system operating conditions. Then, the stacked auto-encoder is used to learn and obtain robust and high-order feature representations. Finally in the off-line training phase, all the representation layers are stacked and a classifier is appended to them. The obtained deep neural network model is then trained with back-propagation in a supervised manner. After the off-line training phase, an online monitoring phase allows the online acquisition of measurements from SCADA in the transmission system. These measurements are fed to the deep neural network, and the results of the classification are used for applications such as situational awareness. A testbed simulating a power grid is used to evaluate the proposed approach (Figure 3.4).

The results show that the introduction of feature learning achieves over 96% in accuracy against three different types of attacks, outperforming the supervised detectors by a small margin. This competitive performance would be beneficial as less details of system model or human expertise is required in constructing

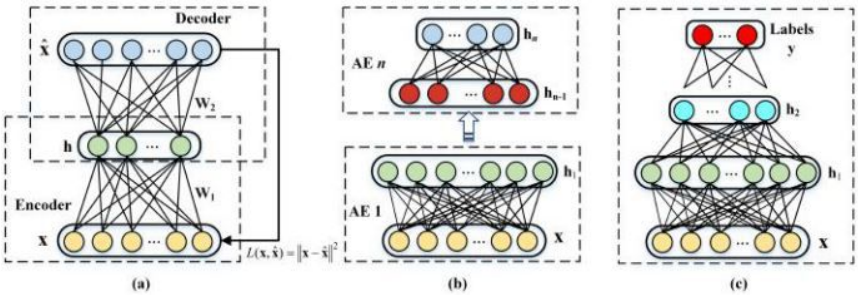

Figure 3.3 – Stacked autoencoders: (a) traditional autoencoder; (b) layer-wise unsupervised pre-training; and (c) supervised fine-tuning
Source: [161] ©2018 IEEE

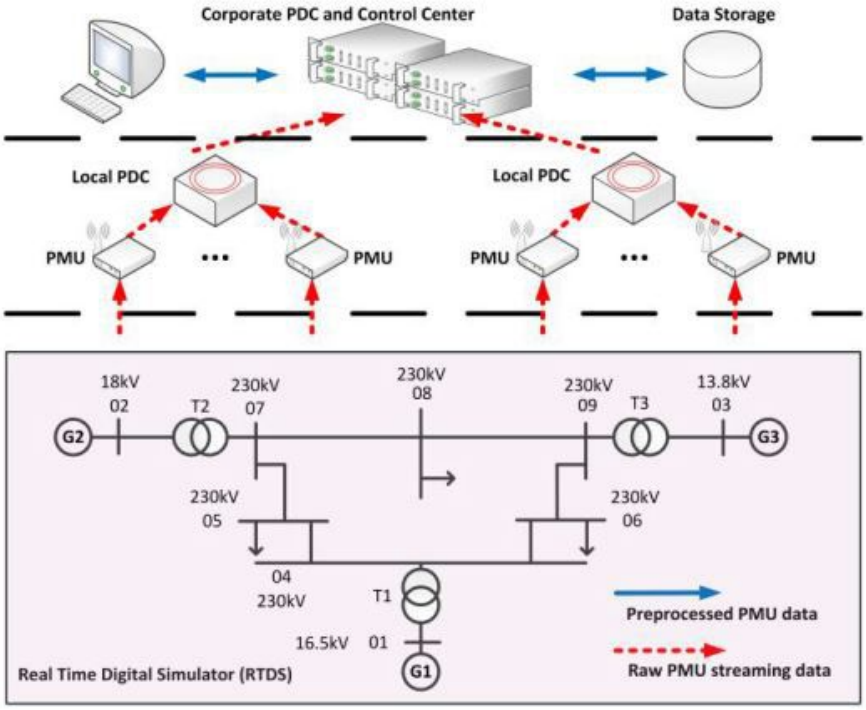

Figure 3.4 – Smart Grid benchmark testbed
Source: [161] ©2018 IEEE

the effective detector. In overall, the proposed framework has the potential to provide adaptive and automatic threat monitoring in complex smart grid applications, as the off-line training deep network model is updated to reflect current situation and used in real-time to detect the network anomalies.

3.3.4 CNN/LSTM anomaly detection in SCADA

The Secure Water Treatment testbed (SWaT) dataset contains up to 36 different cyber-attacks. To evaluate the use of unsupervised feature learning for intrusion detection in such system, [92] proposes two models using either Long Short Term Memory (LSTM) or 1D Convolutional Neural Networks (CNN) as feature learner. They use mean squared error (MSE) as a loss function and AdamOptimizer with weight decay for all experiments. The weight decay as a regularisation technique prevent model overfitting and the AdamOptimizer [88] is computationally efficient and require little memory. The first Deep Neural Network (DNN) architecture is a stacked LTSM with a fully connected layer at the top for classification purpose. With the LSTM model, setting a learning rate between 0.001 and 0.00001, and a decay rate ranged from 0.9 to 0.99 they were able to test various depths of LSTM layers (from 64 to 2048) and sequence lengths (between 50 and 1000). The 1D CNN architecture adopted the classical Convolution-ReLU-MaxPooling scheme, where convolutions are 1D and applied to each feature separately along the time axis. Different kernel size were used for the experimentations. On top of the convolutions layer, a fully connected layer is added for prediction, and dropout is used to prevent overfitting. The authors tested diverse variations of this CNN architecture, by adding a batch normalization layer or by replacing the basic CONV-RELU-POOL block with (CONV - RELU) × N-MAXPOOL architecture. They also replaced the convolutionals layers by Inception layers [147] know to provide superior performance while keeping computational cost low. The Inception layers use sparse network connections instead of the fully connections used by convolution layers, hence the reduction of the computational overhead. The experiments were conducted on the Secure Water Treatment testbed (SWaT) dataset, which represents a scaled-down version of a real-world industrial water treatment plant which has 36 different cyberattacks. The proposed 1 D CNN model successfully detected 32 out of 36 attacks of the SWaT dataset, representing 89 % of detection rate, which is fairly good, but need to be improved.
The comparison of the different architectures shows that LSTMs and inception-based convolution converge the fastest and produce the lowest training error

rate. The anomaly detection algorithm provides high Area Under Curve (AUC), reaching 0.967 for the eight layers convolutional network. The training and testing times of CNN compared to LSTM network were shorter by a factor 10 to 20 for testing and 15 to 40 for training respectively. Concerning the attack detection performance comparison, the authors show that pure CNN networks demonstrated better anomaly detection results than their LSTM alternatives. The proposed CNN has a detection rates reaching 85 % with a 100 % precision.

3.3.5 Conditional Deep Belief Networks for False Data Injection in Smart Grid

As a countermeasure for False Data Injection (FDI) attack for electricity theft in smart grids, [67] proposes a detection mechanism which mainly consists of a State Vector Estimator (SVE) and a Deep-Learning Based Identification (DLBI) scheme. SVE evaluates the quality of the real-time measurement data by calculating the l-2-norm of measurement residual that is compared with a predetermined threshold θ. When the FDI attack bypass the SVE engine, the Deep Learning-Base Identification (DLBI) tries to detect the compromised data. The proposed Deep Neural Network is a Conditional Deep Belief Network (CDBN) that integrates the standard Deep Belief Network (DBN) with Conditional Gaussian-Bernoulli RBM (CGBRBM) (Figure 3.5). CGBRBM is capable of addressing real-valued input and modeling the impact of the historical observations on the current behavior feature extractions. The use of CDBN allow the analysis of temporal attacks patterns that are presented by the real-time measurement data from the geographically distributed sensors/meters [159]. On the other hand, using CGBRBM on the first hidden layer and regular RBM for the other hidden layer reduces the training and execution time of CDBN architectures. The proposed CDBN which is a binary classifier is able to detect unobservable FDI attacks in real-time by learning the temporal behavior features of the FDI attacks. The CDBN is trained in an unsupervised manner and a fully connected layer is added on top of the model with a binary output node which has a sigmoid activation function. The whole deep neural network structure is then fine-tuned with back-propagation supervised training with labeled data.

The proposed CDBN efficiently reveal the high-dimensional temporal behavior features of the unobservable FDI attacks that bypass the SVE mechanism with a high accuracy rate over 94% even in the presence of occasional operation faults, meaning that unknown attacks could be detected.

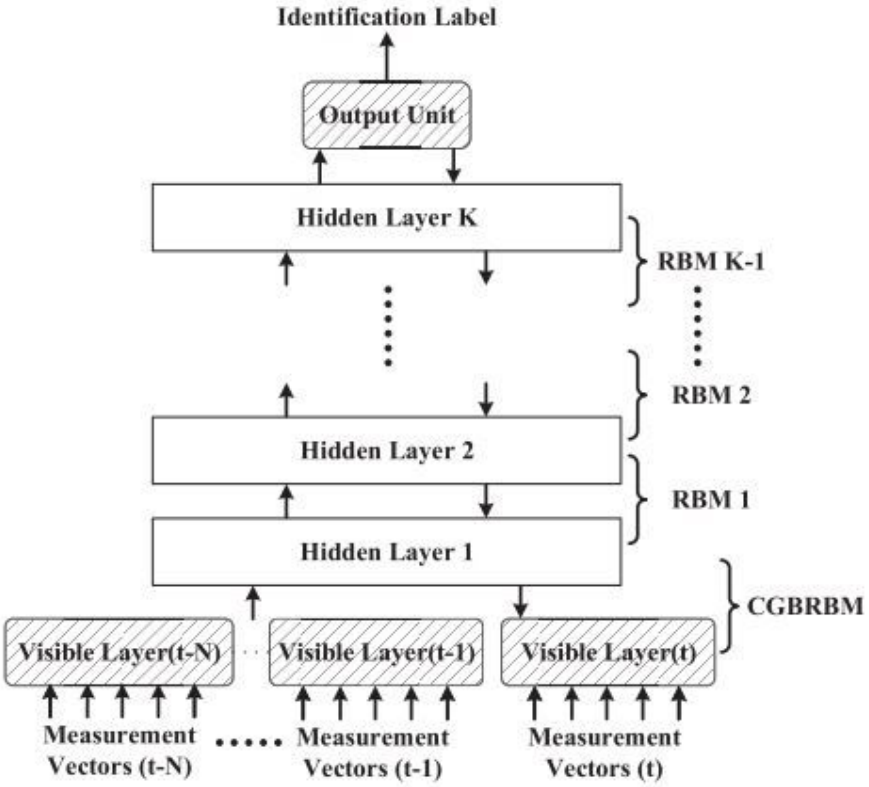

Figure 3.5 – CDBN Architecture
Source: [67] ©2017 IEEE

3.3.6 RBM-based Deep Auto-encoder for Anomaly Detection and Fault Analysis of Wind Turbine Components

Wind turbines usually operate in harsh and variable environment, making theirs components such as gearbox, main bearing, generator, inverter and controller subjet to failure. This situation can lead to unavailability and even destruction, causing expensive repair costs of wind turbines. As a remedy of this situation, the authors [168] present a deep auto-encoder (DAE) approach to detect early

anomalies as well as provide fault analysis of wind turbines components. The data associated to each wind turbine component is extracted in order to build the DAE model. The deep auto-encoder model is used to extract the important features and their relationship from the SCADA data of wind turbine components. The DAE architecture is a deep learning network composed of multiple Restricted Boltzmann Machine (RBM) stacks [59]. The use of a DAE based on RBM building blocks is because of the power of RBM in highly capturing the variational potential of input data [52]. Two major steps are involved in the DAE training process i.e. pre-training and fine-tuning. The pre-training phase, is a layer-wise pre-training of each composing RBM. The pre-training allows the initialization of the deep auto-encoder. During the pre-training phase, the long-term normal operating unlabeled SCADA data is used. Following the pre-training phase which initializes the weights and bias of the DAE is the fine-tuning step. Taking advantage of the SCADA labeled data in long-term normal operation SCADA data, the back-propagation (BP) algorithm is used for a supervised learning to improve the representation of data features and optimize the parameters of hidden layers in the fine-tuning. The SCADA data fed to the DAE is encoded, then decoded, and a reconstruction error is calculated (Figure 3.6). A SCADA data samples obtained from wind turbine normal operation is used to train the DAE model. The training process allow the DAE to extract the internal relationship between the input and the output, and setup the model parameters. Next, an index of component health condition is defined by the reconstruction error of the input and output of the DAE network. For a better monitoring of the index, a dynamic adaptive threshold was proposed on the basis of the detection index calculated by the wind turbine SCADA data. The anomaly detection and fault location analysis is performed by combining the the reconstruction error, the adaptive threshold and the input-output residual. The proposed DAE model is capable of avoiding false alarms and of giving valid warnings at an early stage. In addition, after the DAE model gives an early warning, the possible fault location of the component can be further determined by analyzing the change trends of the SCADA variable residuals of the wind turbine components.

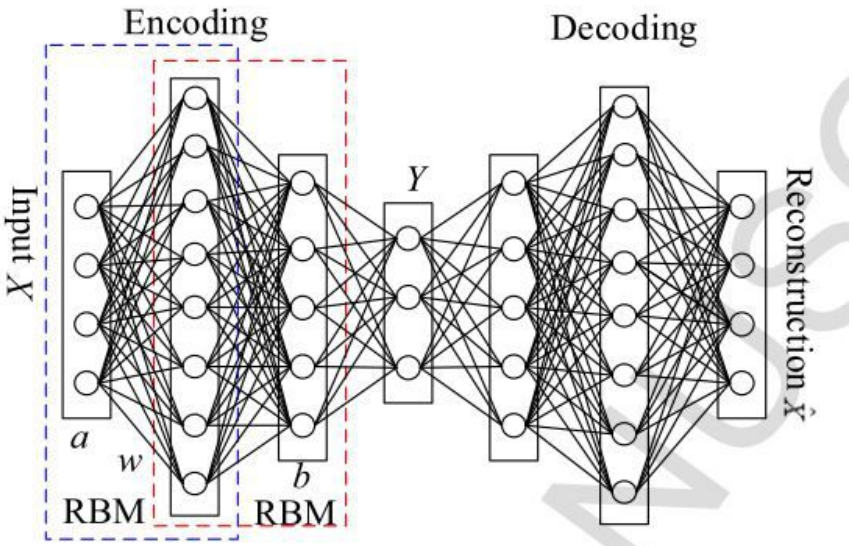

Figure 3.6 – Structure of DAE Network
Source: [168]

3.3.7 Gas Turbine Combustors monitoring with Stacked Denoising Auto-encoder and Extreme Learning Machine

In order to monitor gas turbine combustors' health and detect abnormal behaviors and incipient faults earlier, [163] proposes a deep neural network approach. The proposed model is a Stacked Denoising Auto-encoder (SDAE) [156], to which an Extreme Learning Machine (ELM) [72] is added. The SDAE used for the unsupervised learning of features allow more robust feature learning, even though the input data is noisy. The feature learned from the SDAE are fed to the ELM module for classification purpose. Unlike in other feedforward neural networks, in ELM, connections between input and hidden neurons are randomly generated and fixed, that is, they do not need to be trained. Training consist in finding connections between hidden and output neurons only, which makes ELM training becomes very fast [72]. The only ELM design parameter is the number of hidden neurons. To test the proposed approach, the authors have used seven months of one turbine data containing normal and abnormal data. In order to demonstrate the effectiveness of unsupervised feature learning for combustor anomaly detection, the authors compare classification performance between using the learned features and handcrafted features. The results show

that the deep learned features give significant better classification performance than the handcrafted features (detection rate of 99 % and 96 % for deep learned features and the handcrafted features respectively) .

3.4 Summary of studied approaches

Table 3.1 shows a summary of the different approaches. For each approach we highlight the feature learning architecture, the classifier used to discriminate the data, the types of the attacks detected and the results in terms of accuracy.

Table 3.1 – Summary of Deep Learning Unsupervised Feature Learning in SCADA

Method	Feature Extractor	Classifier	Attacks/Faults	Training time	Accuracy
SVE + CDBN [67]	CDBN-RBM	Fully connected NN	False Data Injection	N/A	>94%
Stacked LSTM + Bloom Filter [50]	Stacked LSTM	Softmax	- Data Injection - Command Injection - Reconnaissance - DoS	35 minutes for 50 epochs	92 %
Stacked AE + Softmax [132]	SAE	Softmax	- DoS - Probe - R2L - U2R	Consume lot of time	- 97.6 % - 86.34 % - 12.98 % - 39.62 %
CNN/LSTM + Fully connected NN [92]	CNN/LSTM	Fully connected NN	Set of 36 different attacks	214 s for 1 epoch	f1-score 92 %
DAE-RBM [168]	DAE	DAE residuals	- Operating anomaly - Faults	0.44 s for 10 min data	Early faults detected
SAE + MLP [161]	SAE	MLP	- Data injection - Remote tripping - Command injection - Relay setting change	High number of features increase computational complexity	96 %
SDAE + ELM [163]	SDAE	ELM	- Faults	N/A	99 %

3.5 Conclusion

This review of SCADA anomaly detection using a deep feature learning approach show that various combinations of deep learning architectures with feature learning capability are used to achieve anomaly detection in SCADA networks. CNN, LSTM, CDBN architectures are used as feature learners, but most of the approaches use a stacked autoencoder approach to learn the salient features. On top of the feature learning architectures, diverse supervised classifiers such as softmax, fully connected neural network, Extreme Learning Machine are used for the classification of the data. The detected anomalies range from false data injection, command injection, reconnaissance attack, DoS, to plant operation anomalies which could be the consequences of cyber-attacks. In most cases, those deep learning based approaches detection rate outperform standard approaches ones. However, deep learning approaches training time remains much higher.

Deep feature learning based approaches are viable for anomaly detection in SCADA systems, even though detection rate, false alarm rate and training time need to be improved.

The next chapter is dedicated to the design of an unsupervised deep neural network feature learning framework for SCADA systems.

Chapter 4

Building an Unsupervised Deep Neural Network Feature Learning Framework for SCADA systems

4.1 Introduction

In the previous chapter, the review of anomaly detection systems using deep learning unsupervised feature learning approaches to protect SCADA networks showed that more and more researches are trying to propose anomaly detection measures using this capability. In fact, the feature engineering is one of the most time consuming in Machine learning, and deep learning have the capability of significantly reduce that time [32]. Moreover, the unsupervised automatic feature learning of deep learning [59] [4] is very important for data-driven anomaly detection systems in SCADA networks because of the heterogeneity, volume and velocity of data in those systems [172] [7].

In this chapter, our objective is to propose a deep learning based architecture that is capable to learn the most important features of a SCADA network data. We are providing the core components of this architecture and its parameters, hyperparameters, activation and loss functions as well as the training algorithm. .

After the presentation of the water storage tank SCADA dataset used for this work in the first section, we give the building process of the stacked sparse denoising auto-encoder in the following section. Thereafter, a section is devoted to the loss function, activation function, parameters and hyper-parameters. In the fifth section, we define the algorithm needed to train the model before the conclusion.

4.2 SCADA Dataset used

The dataset used is obtained from a testbed of the Mississippi State University SCADA Security Laboratory and Power and Energy Research laboratory. The physical system is a water storage tank system [115]. The records of the dataset were captured from the control system of the water storage tank that models oil storage tanks found in industries like chemical or refineries [55]. The physical process is made of two storage tanks (a primary and a secondary one), a pump that moves the water from the secondary storage tank to the primary one, a relieve valve which allows the water to flow from the primary storage tank to the secondary tank, and a sensor which indicates the water level in the primary tank as a percentage of total capacity. The control system of the water storage tank system have three part : an HMI that allows a human operator to monitor and control the water storage tank, a Master Terminal Unit (MTU), a Remote Terminal Unit (RTU) and communication links. A complete description of the system operation can be found in [116].

Dataset records categories

The dataset contains normal records and 28 attacks against the Modbus Industrial Control System that monitor the water storage tank. The attacks types are Naïve Malicious Response Injection (NMRI), Complex Malicious Response Injection (CMRI), Malicious State Command Injection (MSCI), Malicious Parameter Command Injection (MPCI), Malicious Function code Command Injection

Table 4.1 – Dataset records categories

Label	Category	Description
0	Normal	Instance not part of an attack
1	NMRI	Naive Malicious Response Injection attack
2	CMRI	Complex Malicious Response Injection attack
3	MSCI	Malicious State Command Injection attack
4	MPCI	Malicious Parameter Command Injection attack
5	MFCI	Malicious Function code Command Injection attack
6	DoS	Denial-of-Service attack
7	Reconnaissance	Reconnaissance attack

(MFCI), Denial-of-Service (DoS), and Reconnaissance attack (Table 4.1).

Dataset attributes

The datasets contain two types of attributes i.e. the network traffic attributes and the payload attributes. The topology and services of the SCADA systems are relatively static compared to traditional IT. Therefore, network traffic attributes can be used to describe normal traffic patterns and thus detect fraudulent activities. Network traffic attributes include the device address, function code, packet length, packet error control information, and the time interval between packets. The attributes of the payload content provide information about the state of the SCADA system. They are useful for detecting the cause of abnormal behavior in hardware (eg PLCs). These attributes include sensor measurements, control command values and physical process states (see Table 4.2).

4.3 Unsupervised feature learning architecture

As seen in the previous chapter, various deep learning based architecture i.e CNN, LSTM, Conditional Deep Belief Network, stacked autoencoder (SAE) or stacked denoising autoencoders (SDAE) are used for feature learning in SCADA

Table 4.2 – water storage tank attributes

Attribute	Type	Description
command_address	Network	Device ID in command packet
response_address	Network	Device ID in response packet
command_memory	Network	Memory start position in cmd.
response_memory	Network	Memory start position in resp.
command_memory_count	Network	Memory bytes for R/W command
response_memory_count	Network	Memory bytes for R/W response
comm_read_fun	Payload	Value command read func.code
comm_write_fun	Payload	Value command write func.code
response_read_fun	Payload	Value response read func. code
response_write_fun	Payload	Value response write func.code
sub_function	Payload	Value of sub-function code
command_length	Network	Total length of command packet
response_length	Network	Total length of response packet
HH	Payload	Value of HH setpoint
H	Payload	Value of H setpoint
L	Payload	Value of L setpoint
LL	Payload	Value of LL setpoint
control_mode	Payload	Automatic, manual or shutdown
control_scheme	Payload	Manual mode compressor/pump
pump_state	Payload	Compressor/pump state
crc_rate	Network	CRC error rate
measurement	Payload	water level
time	Network	Time interval betw. 2 packets
label	Provided	Manual classification

networks anomaly detection. Stacked autoencoders can use standard autoencoders or Restricted Boltzmann Machine (RBM) as building blocks. However, Tan and Eswaran proved in a study [148] that stacked autoencoder using RBMs tends to make the network more focused on the training dataset, resulting in a low generalization. A comparison of performances of the stacked denoising autoencoder (SDAE) introduced by Pascal Vincent and his team [156] and other deep neural networks based architectures such as on Multi Layer Perceptron (MLP), Deep Belief Network (DBN) corresponding to stacked Restricted Boltzmann Machine (RBM), stacked autoencoder (SAE) shows SDAE architecture achieves the best performance in terms of test error rate.

Therefore, the proposed deep feature learning architecture will be based on the stacked denoising autoencoder (SDAE) architecture. However, instead of using a simple denoising autoencoder (DAE) as building block, we are introducing a sparsity parameter to allow more robust feature learning.

4.4 Sparse Denoising Autoencoder (SpDAE)

4.4.1 Auto-encoders

An Auto-encoder is a Neural Network that is trained to try to reproduce an approximation of its input (Figure 4.1). But merely learning the identity function is not sufficient for learning representation of the input; thus, a traditional approach to combat the reproduction of the identity function is to use a bottleneck to produce a under-complete representation where the dimension of the hidden layer is less that the input dimension [155]. A basic auto-encoder has an input layer, a hidden layer and an output layer. The first part of the network formed by the input and the hidden layer is the encoder and likewise, the second part formed by the hidden layer and the output layer is the decoder. The encoder tries to represent the input layer at the hidden layer while the decoder reconstructs the input to its original dimension at the output layer [127]. Let's set input $x = (x_1, x_2, .., x_n)$, hidden layer $h = (h_1, h_2, ..., h_d)$ and output $y = x = (y_1, y_2, \ldots, y_n)$. Let's $W \in \mathrm{R(nxd)}$ *and* $\mathrm{b} = (\mathrm{b}_1, b_2, \ldots, b_d)$ be the weights and bias at the hidden

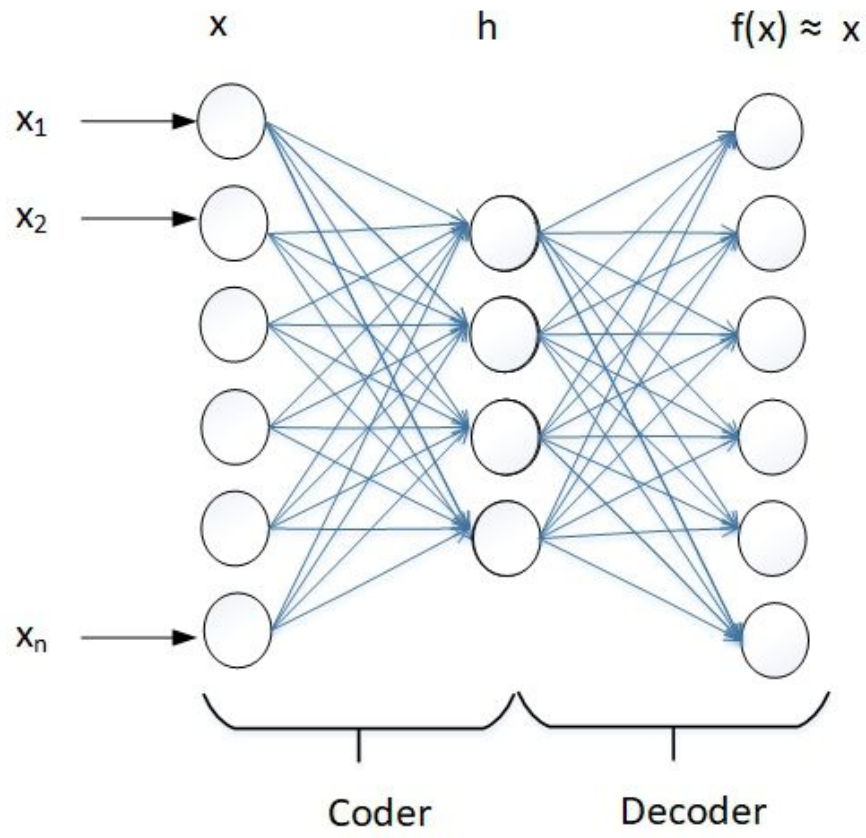

Figure 4.1 – Auto-encoder

layer and $W' \in$ R(dxn) *and* b' $= (b'_1, b'_2, \ldots, b'_n)$ be the weights and bias for the output layer. The output of the hidden layer can is $h = f1(Wx+b)$ and the output of the output layer id $\hat{y} = f2(W'h+b')$, where $f1$ and $f2$ are activation functions (Linear, Sigmoid, ReLU,...). Let's define the cost C function.

$$C = \sum_{k=1}^{m} \left\| \hat{y}^{(k)} - y^{(k)} \right\|_2^2 = \sum_{k=1}^{m} \left\| \hat{y}^{(k)} - x^{(k)} \right\|_2^2$$

The cost function C can be minimized [59] based on training data by deriving the model parameters (W, W', b, b').

$$\hat{\theta} = \arg\max_{\theta} C(\theta) = \arg\max_{\theta} \sum_{k=1}^{m} \left\| \hat{y}^{(k)} - x^{(k)} \right\|_2^2$$

When using a gradient descent optimizer, the learning rule at iteration t is defined by :

$$\theta^{(t+1)} = \theta^{(t)} - \epsilon \Delta_\theta C(\theta^{(t)})$$

Where ϵ is the learning rate and $\Delta_\theta C(\theta^{(t)})$ is the gradient of the cost function with respect to θ at $\theta = \theta(t)$. Beside the under-complete approach to learn interesting feature from an input, two novel strategies are used to get more meaningful representation of input data i.e. the sparsity parameter and denoising.

4.4.2 Sparse Auto-encoder

As previously stated, in under-complete auto-encoders, the number of hidden neurons is less than the number of input features. In this way, the encoder is forced to learn the most interesting representation of the input. Another way to learn interesting features is to introduce a sparsity constraint in the hidden layer. A neuron is "active" when its output value is close to 1 and "inactive" when the output value is close to 0. The sparsity will constrain the neurons to be inactive most of the time [122]. If $a_j^{(2)}$ is activation of hidden unit j in the auto-encoder, $a_j^{(2)}(x)$ is the activation of this hidden unit for an input x. The average of the activations for the hidden unit j for a training set of m samples is

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} [a_j^{(2)}(x^{(i)})]$$

The idea is to force the average of the activations to be very close to 0. This is done by introducing a sparsity parameter ρ that would enforce the constraint $\hat{\rho}_j = \rho$. Typically, ρ is a small value close to 0($\rho = 0.05$). To satisfy the constraint, the hidden unit's activation must mostly be close to 0. A penalty term is introduced to penalize deviating from ρ, using the Kullback-Leibler (KL) Divergence.

Kullback-Leibler (KL) Divergence

The KL-divergence [122] [127] is a function that measures how different two

Bernoulli random distributions are.

$$KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

Where ρ and $\hat{\rho}_j$ representing the means of two Bernoulli random variables.
If we set the value of ρ very small (0.05 for example), and $\hat{\rho}_j$ being the mean of the activation of a given node over a the dataset, that will cause the node to rarely activate, thus introducing the sparsity.

4.4.3 Denoising auto-encoders

To avoid simply copying the input and guaranty learning useful representation from the input, another strategy apart from the sparsity has been introduced, namely, the denoising auto-encoder. This strategy aims to clean the partially corrupted input or denoising the input. The denoising process which is defined as a training criterion for learning to extract useful features allows a more robust feature extraction of the input [155] [156]. For Denoising autoencoders (DAE), we first corrupt the initial input x into $\tilde{x}$ by means of a stochastic mapping $\tilde{x} \sim q_D(\tilde{x}|x)$. Then, as in basic auto-encoder, a representation of the corrupted input is evaluated at the hidden layer $y = f_\theta(\tilde{x}) = s(W_{\tilde{x}} + b)$, and finally, we reconstruct back the corrupted input $z = g_{\theta'}(y)$. We then train the parameters θ and θ' to minimize the average reconstruction error over a training set. z must be as close as possible to the uncorrupted input x.

4.4.4 Sparse Denoising auto-encoder

The sparse denoising auto-encoder is a denoising autoencoder with a sparse parameter. The denoising autoencoder, instead of merely minimizing the loss function

$$L(x, g(f(x)))$$

where $f(x)$ is the coder and $g(f(x))$ the decoder, minimizes

$$L(x, g(f(\tilde{x})))$$

where $\tilde{x}$ is a copy of x that has been corrupted by some form of noise.
The sparse autoencoder on the other hand adds a sparsity penalty $\Omega(h)$ on the code layer h in addition to the reconstruction error [59] :

$$L(x, g(f(x))) + \Omega(h)$$

where $g(h)$ is the decoder output and $h = f(x)$, the encoder output. The sparsity term of an autoencoder make it respond to unique statistical features of the dataset it has been trained on, rather than simply acting as an identity function.

The sparse denoising autoencoder will then minimize the loss function

$$L(x, g(f(\tilde{x}))) + \Omega(h)$$

4.4.5 Activation Function

In chapter 2, we saw that the ReLU activation function ($f(x) = max(0, x)$) is computationaly efficient compared to the Sigmoid and Tanh and moreover, was solving the Vanishing Gradient Problem. However, it also suffers from the Deying ReLU problem which causes neurons to not fire. So, we are choosing to use the Leaky ReLU in the hidden layers of the stacked sparse denoising auto-encoder. It has the ReLU advantages, but fixes its Dying ReLU problem.

$$f(x) = \begin{cases} 0.01 & x < 0 \\ x & x >= 0 \end{cases}$$

4.4.6 Loss Function

Each auto-encoder of the stacked sparse denoising auto-encoder is trained separately so as to reconstruct its input. If m is the number of samples in the training set, we will use the mean square error as the loss function [122]

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}(h_{W,b}(x^{(i)}) - x^{(i)})^2)$$

where $\theta = \{W, b\}$

As there is a sparsity parameter, we should adjust the cost function accordingly, which becomes

$$J_{sparsity}(\theta) = J(\theta) + \beta\Omega_{sparity}$$

Where $\Omega_{sparsity}$ is the sparsity term allowing only a small number of neurons of the hidden layer h to be active at each iteration.

$$\Omega_{sparsity} = \sum_{i=1}^{k} KL(\rho||\hat{\rho}_i) = \sum_{i=1}^{k}\left(\rho\log\frac{\rho}{\hat{\rho}_i} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_i} \right)$$

Where k is the number of neurons of the hidden layer. The overall cost function is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}(h_{W,b}(x^{(i)}) - x^{(i)})^2) + \beta\sum_{i=1}^{k}\left(\rho\log\frac{\rho}{\hat{\rho}_i} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_i} \right)$$

β is a factor allowing to adjust the relative importance of the sparsity term in the loss function.

4.5 Stacked Sparse Denoising Autoencoder (SSpDAE)

4.5.1 Process of building the SSpDAE

In the process of stacking the sparse denoising auto-encoders, the input of the first auto-encoder is the SCADA dataset features, and the hidden layer is h_1 with some number of nodes. The decoder part of the first auto-encoder is discarded. For the second auto-encoder, the input layer is the hidden layer h_1 of the first auto-encoder, and the hidden layer of the second auto-encoder is h_2. We do this process until we reach the n^{th} auto-encoder which input is the hidden layer h_{n-1}

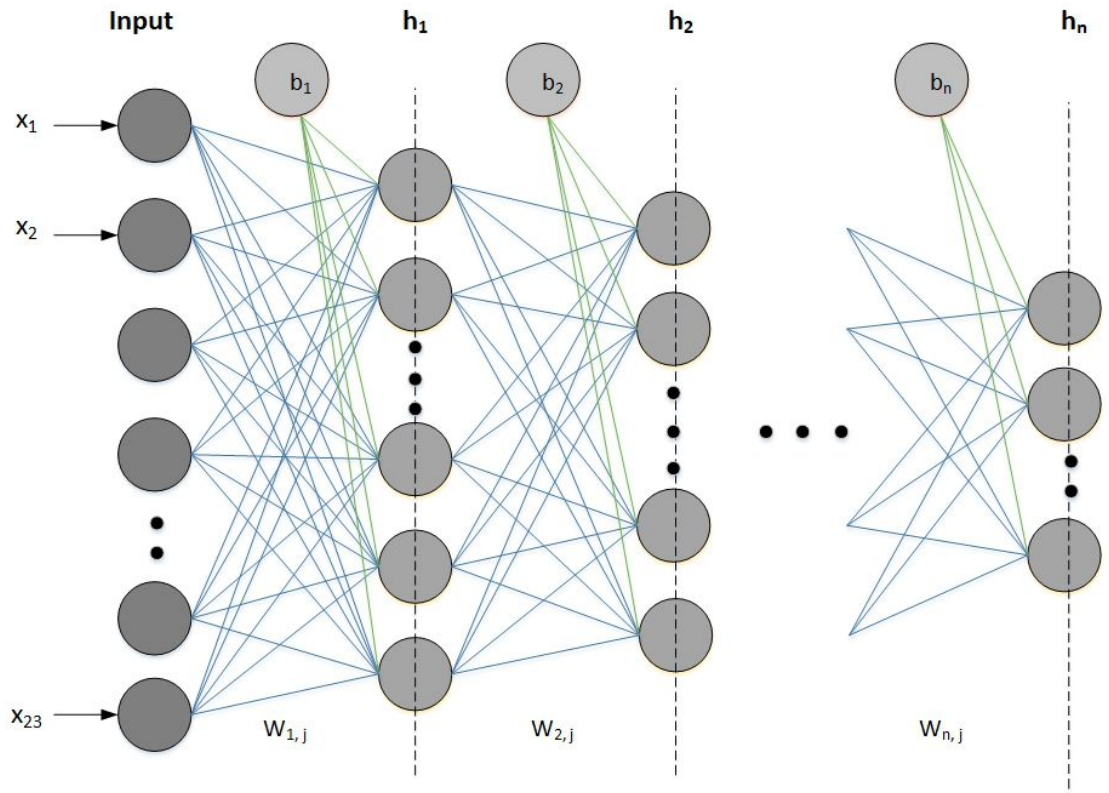

Figure 4.2 – Stacked Sparse Denoising Auto-encoder

of the $(n-1)^{th}$ auto-encoder, and the hidden layer of the n^{th} auto-encoder is h_n (Figure 4.2).

Now that the stacked denoising auto-encoder global structure is defined, in the next section, we will define the loss function, the activation function, parameters and hyper-parameters like the number of layer, the number of nodes per layer and the regularization used to avoid overfitting.

4.5.2 Depth and width of the SSpDAE

The number of layers (depth) might be set carefully, as low latency is an essential requirement in SCADA networks. The deeper the neural network gets, the longer it takes to train the DNN model. Therefore, trade-offs should be made between the detection rate and the training time of the model. Unlike in Convolutional

Neural Networks where the deeper the network is, the better it performs, with deep neural networks, it is shown in practice that 3 layer Neural Network will outperform a 2 layer one, but going deeper rarely helps much more [74]. Therefore, we will set the number of layers to 2, and vary it making the architecture deeper in the implementation phase in chapter 6. With the use of the sparsity parameter in the model, we will set a number of nodes for the hidden layer higher than the input dimension.

4.6 Conclusion

In this thesis, we have used a water storage tank testbed SCADA dataset that have normal and anomalous records. The attributes of the dataset are either from the network or measures from the physical process observations. The dataset contains normal as well as attacks records. There is an overall 28 attacks grouped into 7 categories which are Naïve Malicious Response Injection (NMRI), Complex Malicious Response Injection (CMRI), Malicious State Command Injection (MSCI), Malicious Parameter Command Injection (MPCI), Malicious Function code Command Injection (MFCI) and Reconnaissance.

As a basis of the proposed architecture, we choose the stacked denoising autoencoder (SDAE) based on its performance in prior studies, to which we added a sparse parameter to make the feature learning process more robust.

Afterwards, we showed the process of building the stacked sparse denoising autoencoder (SSpDAE), where the first sparse denoising autoencoder (SpDAE) input is the SCADA data attributes, and for the subsequent layers, the output layer of a previous SpDAE is discarded, and its hidden layer is the input layer of the current SpDAE. Leaky ReLU is used as activation function for its computational efficiency and its non-dying neurons capabilities. The loss function for each SpDAE is a mean square error to which we added a weight decay for regularization and a sparsity parameter to control the number of firing neurons per layer. Finally, as low latency is essential in SCADA networks, in the implementation phase, trade-offs should be made between the detection rate

and training time of the model.

In the next chapter, we will use the SSpDAE to build an hybrid deep neural network anomaly detection system that is leveraging the unsupervised feature learning capability of deep learning for anomaly detection in SCADA systems. A distributed approach of the proposed anomaly detection system is also proposed to deal with the high training time of deep architectures.

Chapter 5

Hybrid Deep Neural Network Anomaly Detection System for SCADA Networks

5.1 Introduction

An anomaly detection system is a system which is able to discriminate normal data from anomalous ones. In order to get best classification results, data with good features have to be presented to the input of the classifier. In classic Machine Learning, the features are handcrafted before the training and classification process. However, Deep Learning has the capability to automatically learn important features of data in an unsupervised manner [20] [59] [105]. In the previous chapter, we have designed a stacked sparse denoising auto-encoder for SCADA networks data unsupervised feature learning.

In the present chapter, we propose a deep neural network for anomaly detection in SCADA systems by adding a supervised classification layer and providing the training algorithms. Furthermore, we provide the design framework of the anomaly detection system and a distributed approach to lessen the model training time.

In the first section, we give details of the design principles of the deep neural network. Afterward, we design the general hybrid anomaly detection system framework. In the following section, we explain the training process of the unsupervised feature learning component i.e. the greedy layer-wise pre-training. This section is followed by the training process of the supervised layer and the fine-tuning of the whole network, as well as the detection phase with test data. Finally, we propose a distributed approach of the hybrid deep neural network anomaly detection system as a way to speed-up the training process.

5.2 Hybrid SCADA DNN Anomaly Detection System

The anomaly detection system we propose has two parts: a data pre-processing part and the anomaly detection part. The proposed anomaly detection approach is an hybrid SCADA deep neural network anomaly detection system, as it uses an unsupervised feature learning engine and a supervised classification engine (Figure 5.1). Before reaching the anomaly detector, the SCADA data is pre-processed inside the data pre-processing engine which includes a data normalization module, a data splitting module, and a data balancing module.

5.3 SCADA Datasets Pre-processing

All the values of the dataset are numerical. The pre-processing encompasses four different steps i.e. Min-Max normalization, dataset splitting into training, validation and test sets, balancing the training and validation sets, and one-hot encoding all the datasets.

5.3.1 Min-Max Normalization

Standardization and Min-Max normalization are two widely normalization techniques in machine learning. However, while the standardization center each feature value on 0, producing values in range [-1, 1], the Min-Max on the other

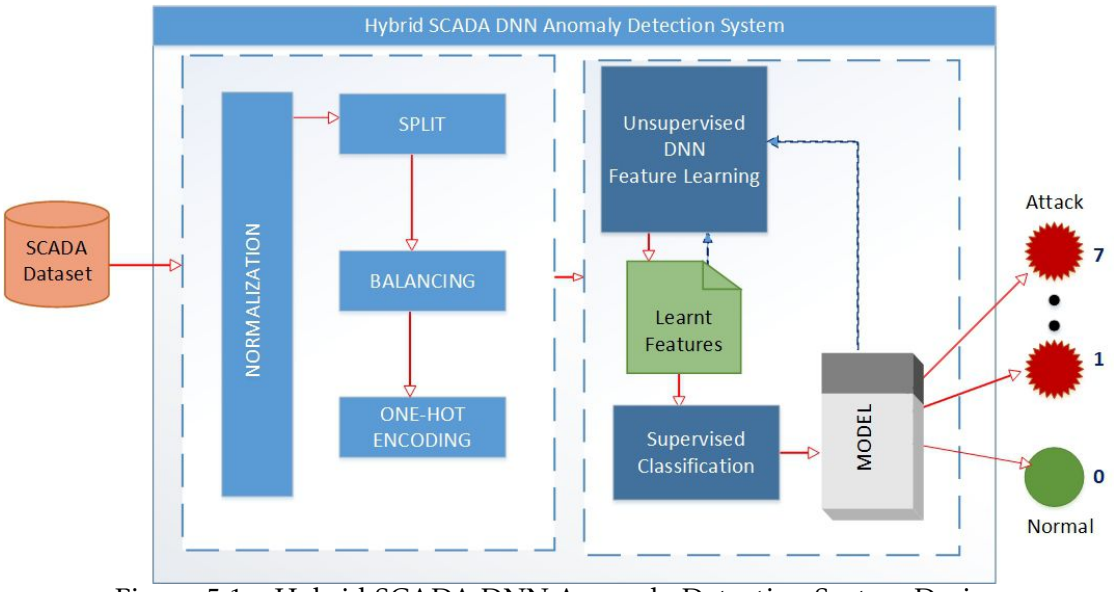

Figure 5.1 – Hybrid SCADA DNN Anomaly Detection System Design

hand produces values in range [0, 1] [128]. As the SCADA dataset contains only positive values, we choose Min-Max as the normalization technique.

The Min-Max normalization consists in scaling all features values of the dataset in range [0,1]. For a feature F with a value x, the normalized value x' is

$$x' = \frac{x - min(F)}{max(F) - min(F)}$$

where $min(F)$ and $max(F)$ are minimum and maximum F values respectively [78].

5.3.2 One-hot encoding

One-hot encoding is a vector representation, where we have a group of bits for which only a single column's value in the vector can have the value 1. All of the other column values will be 0. One-hot encoding is often used to represent categorical features [128].

5.3.3 Data Splitting into Training, Validation and Testing Sets

To split the datasets, we use a cross-validation technique. The two most commonly used cross-validation techniques are the hold-out cross-validation and the k-fold cross-validation [133]. The hold-out cross-validation is a widely-used cross validation technique that is efficient and easy to use. The dataset is separated into three mutually disjoint subsets, i.e., a training, a validation and a testing sets. There is no restriction on the size of the three data subsets, making this technique suited for big datasets. The model is trained on the training set and the validation one is periodically used to evaluate its performance to avoid overfitting. The training is stopped when the performance on the validation set is good enough or when it stops giving better results.

With the k-fold cross validation technique, the dataset is divided into k parts of the same size. One part acts as the validation (testing) set, while the others form the training set. The process is repeated for each of the k parts. The k-fold cross-validation is useful when not enough data is available for an hold-out cross-validation. We are using the hold-out cross-validation in the experiments.

5.3.4 Dataset Balancing

Many approaches exist for unbalanced datasets balancing, e.g. random oversampling, random undersampling, the Synthetic Minority Oversampling Technique (SMOTE) [27] or the Adaptive Synthetic (ADASYN) [65] sampling methods. Oversampling methods balance training dataset by increasing the number of minority class example, while undersampling methods balance training dataset by decreasing the number majority class examples. In section 6.5 of the next chapter, the dataset distribution shows that some minority class has very few samples compared to normal instances. We then focus the balancing method on oversampling, SMOTE, and ADASYN. Tests results with the SCADA dataset give better results when applying random oversampling.

5.4 Hybrid SCADA DNN Anomaly Detection Engine

The main part of the system design is the anomaly detection engine (Figure 5.2). This engine is made of an unsupervised deep neural network SCADA feature learning module and a supervised classification module.

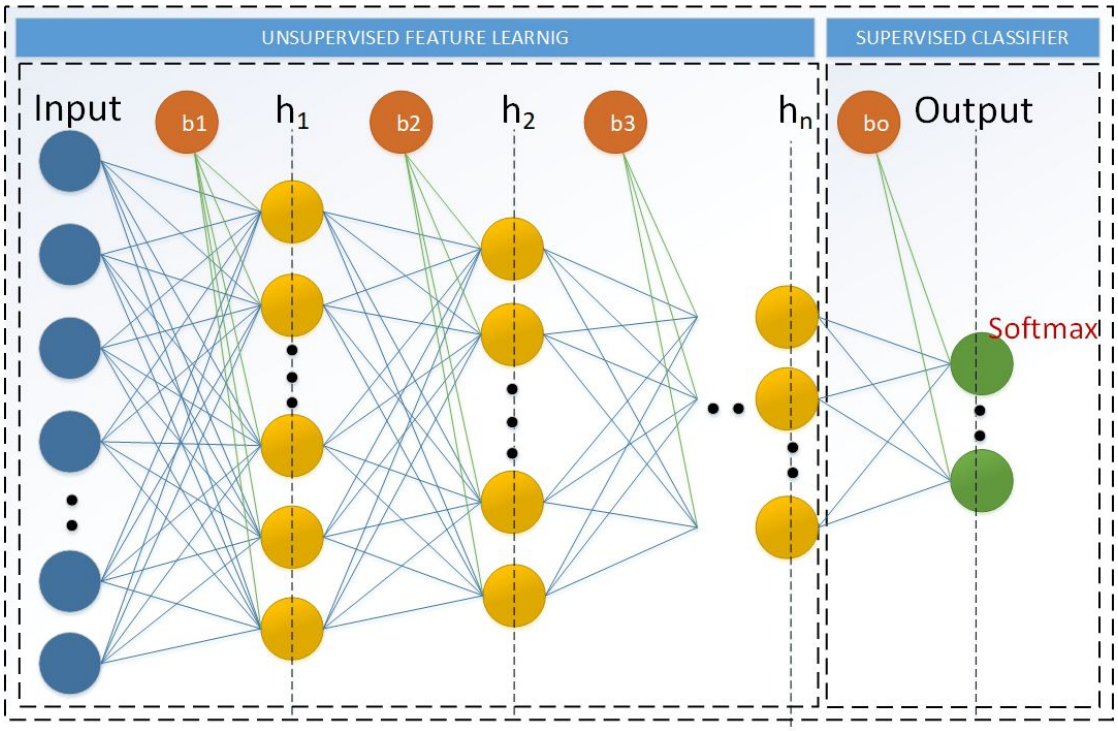

Figure 5.2 – Hybrid SCADA DNN Anomaly Detection System Engine

5.4.1 Unsupervised Deep Neural Network SCADA Feature Learning Module

In the previous chapter, we have designed an unsupervised feature learning module to be used in the deep neural network-based anomaly detection system. This module is a stacked sparse denoising auto-encoder i.e a stacked auto-encoder with sparsity and denoising parameters (Figure 5.3). The purpose of this module is to learn the most important features of the SCADA unlabeled data, so that those features will be used subsequently in a classifier to discriminate data between normal or abnormal classes.

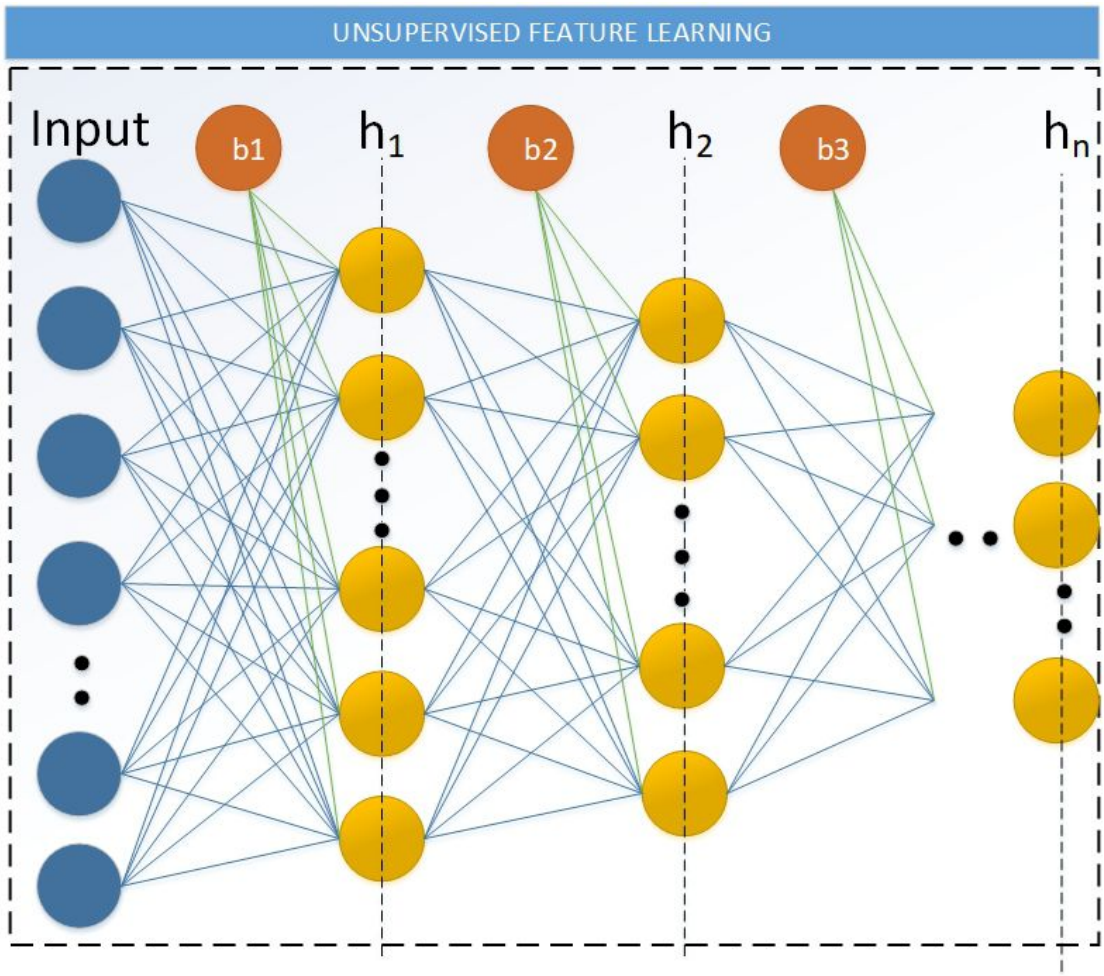

Figure 5.3 – Unsupervised Feature Learning

5.4.2 Supervised Classification Module

The SCADA dataset has records from normal operation as well as those caught from seven different categories of attacks. There is an overall eight categories of records in the dataset. Hence, we are solving a multiclass classification problem which consists in discriminating the different records of the dataset with respect to their correct category.

A layer with nodes using a softmax activation function is suited for multiclass classification problems [45] [127]. Therefore, in the proposed architecture, we use on top of the unsupervised deep feature learner, a supervised classifier which is a softmax layer (Figure 5.4).

When using a softmax layer for a multiclass modeling problem, we only care about the best score across these classes, and we use an arg-max() function of the softmax output layer to get the highest score of all the classes. The softmax output layer gives us a probability distribution over all the classes [128].

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_i}} \quad for \quad j = 1, ..., k$$

If we have a weight matrix W and a bias b, the probability that an input vector x is a member of a class i, a value of a stochastic variable Y , can be written as [59] :

$$P(Y = i|x, W, b) = softmax_i(Wx + b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

The model's prediction y_{pred} would be the class which has the highest probability :

$$y_{pred} = argmax_i P(Y = i|x, W, b)$$

5.4.3 Loss Function of the Supervised Classifier

The loss function that will be defined for the supervised classifier training will also be used to train the hybrid deep neural network.

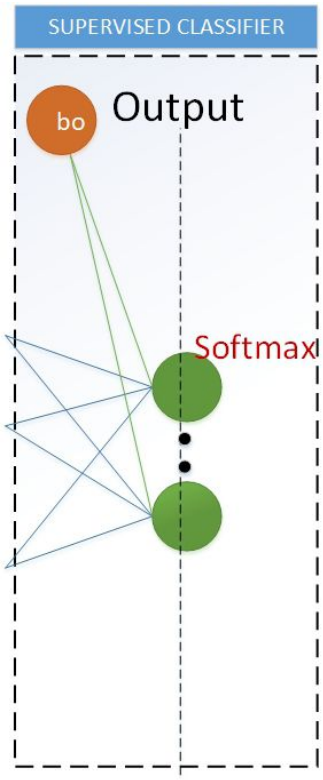

Figure 5.4 – Supervised Classifier

The supervised classifier defined in the previous section is a softmax layer which output is a probability distribution over all the classes. The input data of the supervised classifier and the hybrid DNN use labeled data. In the section 5.3 concerning the SCADA datasets pre-processing, those labels have been one-hot encoded. This is actually a probability distribution where the actual class value is 1 and all the others 0. We then need a loss function capable of comparing two probability distribution (the actual label of data one-hot encoded and the outcome of the classification output by the softmax function). In the sub section 2.3.4 of the chapter 2, we show that cross-entropy is the loss function used to measure the similarity between two probability distributions P and Q:

$$H(P,Q) = -\sum_{x \in X} P(x) log Q(x)$$

Therefore, the training process of the supervised layer as well as the whole hybrid deep neural network will consist in minimizing a cross entropy loss function.

5.5 Training the SCADA Hybrid Anomaly Detection System

There are three steps in the training of the SCADA hybrid anomaly detection system i.e., the greedy layer-wise unsupervised pre-training step, the softmax supervised layer training step, and the entire network fine-tuning step which is a supervised training.

5.5.1 Greedy Layer-wise Unsupervised Pre-training Step

The unsupervised pre-training or greedy layer-wise unsupervised pre-training [15] [71] [59] [12] plays a key historical role in the revival of deep learning enabling the training of deep supervised networks without requiring architectural specializations like convolution or recurrence. This approach relies in on a single-layer representation learning such as a RBM, a single autoencoder, a sparse coding model, etc. Each layer is pre-trained using unsupervised learning. A new representation of the date is produced from the output of the previous layer.

Taking the example of a stacked autoencoder, the greedy layer-wise pre-training is called *greedy* because it uses a greedy algorithm i.e. it optimizes each piece of the deep network (the autoencoder) independently, one at a time, rather than optimizing all the pieces altogether. The term *layer-wise* is because the independent pieces are the network layers. The k-th layer is trained while the previous layer of the deep network are maintained fixed. The training is *unsupervised* because an unsupervised learning algorithm is used. Finally, it is a *pre-training* because, the greedy layer-wise unsupervised pre-training is a first step before the training of the whole network called *fine-tuning*, where all the layer are trained together. The greedy layer-wise unsupervised pre-training can be used as an initialization

strategy for deep networks [71].

For the training process of the stacked sparse denoising autoencoder, the first Auto-encoder is trained, and the coded representation of the input i.e. h_1 is used as the input of the second auto-encoder. We train the second auto-encoder and likewise, another abstraction of the input features is the second hidden layer h_2. Finally, in the same way, the n-th auto-encoder is trained, and a final abstraction of the input feature is represented by the *layer-wise*-th hidden layer h_n.

Algorithme 2 : DNN_Unsupervised_FL

Data : SCADA_dataset
Result : W, b, learnt_features
initialize (W, b, W', b')
`/* L = number of hidden layers */`
for $l \leftarrow 1$ **to** L **do**
  foreach $example \in SCADA\_dataset$ **do**
    $H \leftarrow f(WX + b)$ `// f is the Leaky ReLU activation`
    $X' \leftarrow f(W'H + b')$
    `/* Minimize`
      $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}(h_{W,b}(x^{(i)}) - x^{(i)})^2) + \beta\sum_{i=1}^{k}\left(\rho \log\frac{\rho}{\hat{\rho}_i} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_i} \right)$ `*/`
    $W, b \leftarrow Minimize\ J(\theta)$
  end
  $learnt\_features \leftarrow H_L$
end

5.5.2 Softmax Supervised Layer Training Step

The supervised classification module is a softmax layer. The transfer function for each node of is a softmax function, which maps each output to a vector of probabilities. The softmax layer has eight nodes corresponding to the number of the different types of the dataset records. This layer is appended to the stacked sparse denoising autoencoder (Fig 5.6). After the greedy layer-wise unsupervised

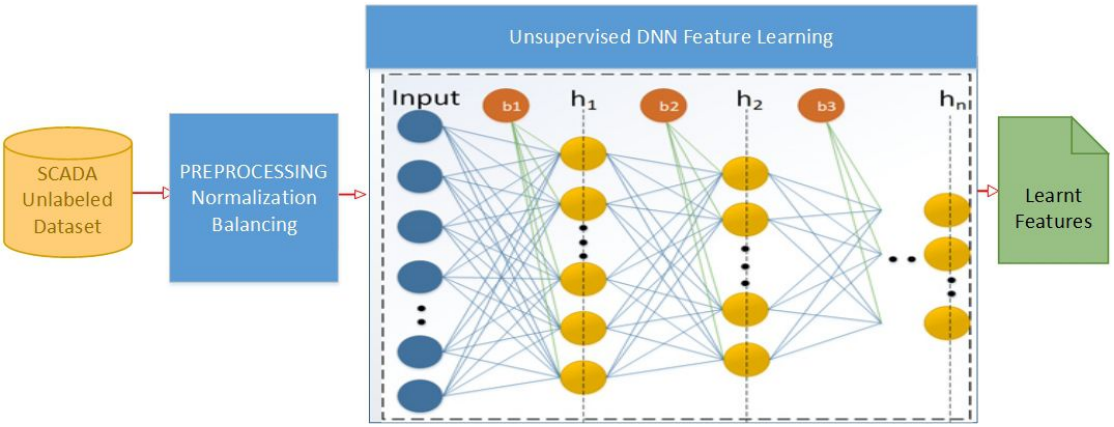

Figure 5.5 – Feature Learning

pre-training of the stacked sparse denoising autoencoder, in this step, we train the Softmax layer with the SCADA training dataset with training labels.

Algorithme 3 : Softmax_Supervised_Classification

Data : network, SCADA_dataset, true_labels, training_epoch

Result : W, b, W_s, b_s

initialize (W, b, W_s, b_s)

for $i \leftarrow 1$ **to** $training\_epoch$ **do**

$W, b \leftarrow DNN\_Unsupervised\_FL$ (SCADA_dataset)

$X, \leftarrow SCADA\_dataset$

$H_1 \leftarrow f(W_1 X + b_1)$ // f is the Leaky ReLU activation

for $l \leftarrow 2$ **to** L **do**

$H_l \leftarrow f(W_l H_{l-1} + b_l)$

end

/* $P(Y = i|x, W, b) = softmax_i(Wx + b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$ */

$pred\_labels \leftarrow softmax(H_L)$

/* Minimize Cross-Entropy $H(P, Q) = -\sum_{x \in X} P(x) log Q(x)$ */

$W_s, b_s \leftarrow Minimize\ D(true\_labels, pred\_labels)$

end

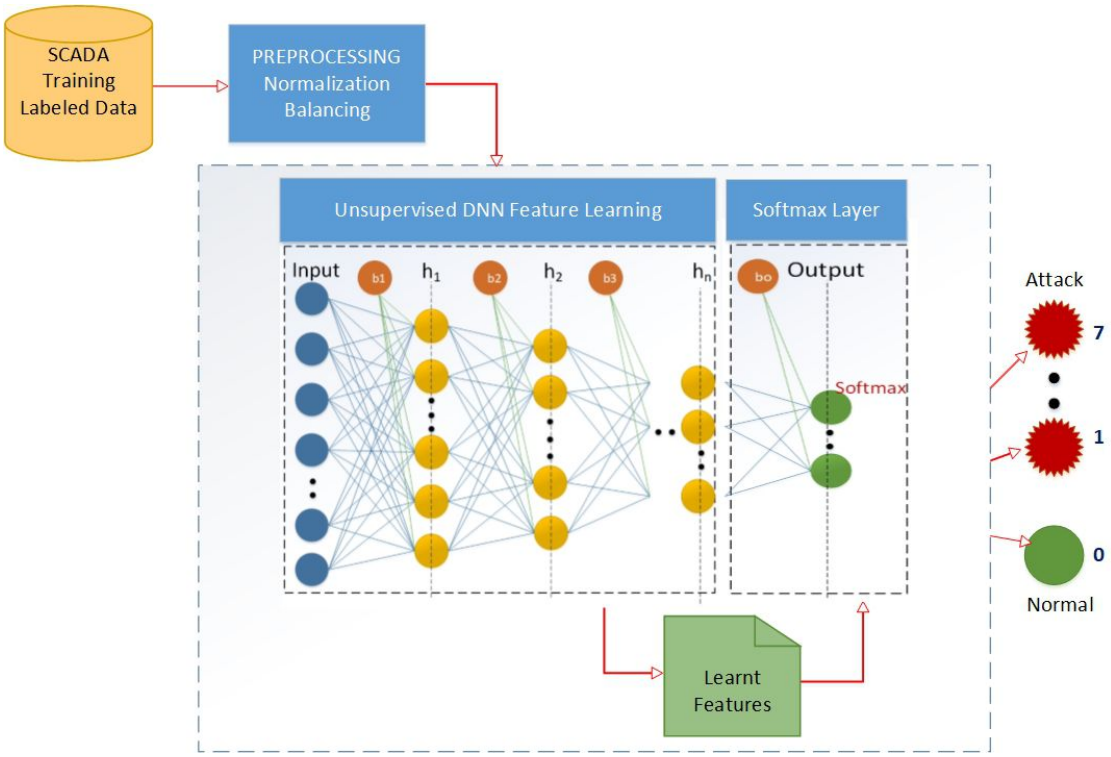

Figure 5.6 – Softmax Layer Training

5.5.3 Hybrid Anomaly Detection System Fine-tuning Step

After the unsupervised feature leaning and the training of the classification layer, the Deep Neural Network parameters i.e. weights, biaises are set. In the fine-tuning [71] [59], the whole network is trained using those parameters and the training dataset with labels. The fine-tuning (Figure 5.7) enables the whole network to be further optimized by gradient descent in order to minimize the

reconstruction error [15], thus increases the classification accuracy .

Algorithme 4 : DNN_Fine_Tuning

Data : $network, SCADA\_dataset, true\_labels, training\_epochs, W, W_s, b, b_s$

Result : W, W_s, b, b_s

for $i \leftarrow 1$ **to** $training\_epoch$ **do**

$X \leftarrow SCADA\_dataset$

$H_1 \leftarrow f(W_1 X + b_1)$ `// f is the Leaky ReLU activation`

for $l \leftarrow 2$ **to** L **do**

$H_l \leftarrow f(W_l H_{l-1} + b_l)$

end

$pred\_labels \leftarrow softmax(H_L W_s + b_s)$

`/* Minimize Cross-Entropy` $H(P,Q) = -\sum_{x \in X} P(x) log Q(x)$ `*/`

$W, W_s, b, b_s \leftarrow Minimize\ D(true\_labels, pred\_labels)$

end

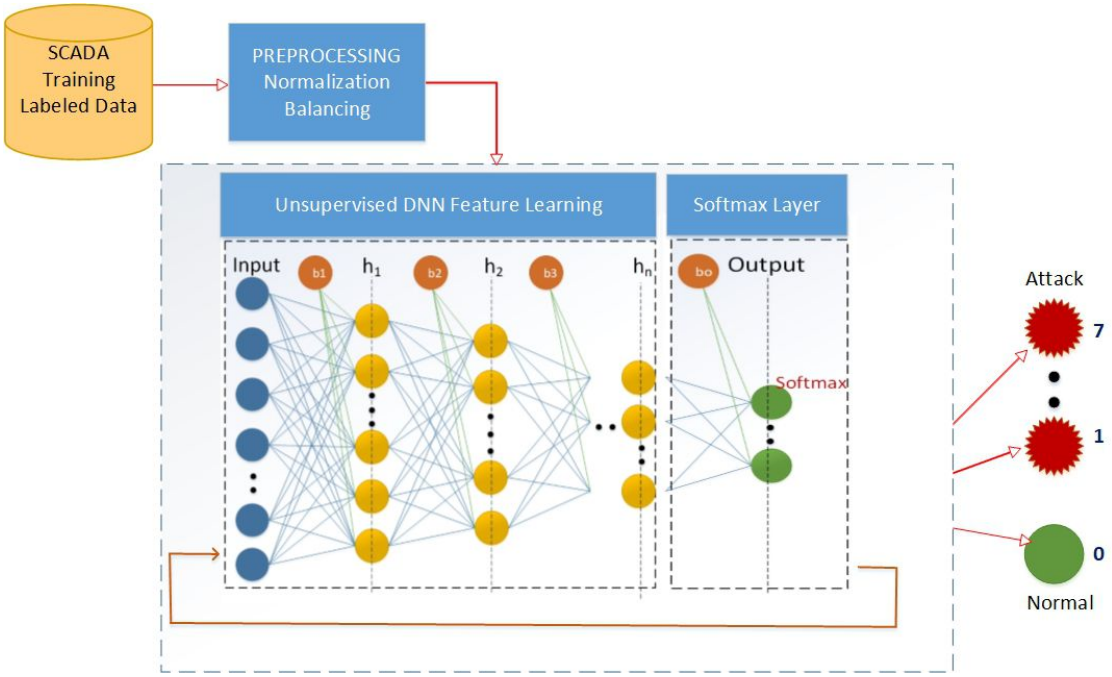

Figure 5.7 – Fine Tuning

5.6 Hybrid Anomaly Detection System Detection

Once fine-tuned, we use the test dataset with labels to evaluate the detection performances of the Deep Neural Network-based Anomaly detection System of SCADA networks (Fig. 5.8).

Algorithme 5 : DNN_Detection

Data : $network, SCADA\_test\_dataset, W, b, W_s, b_s$

Result : $pred\_labels$

$X \leftarrow SCADA\_test\_dataset$

$H_1 \leftarrow f(W_1 X + b_1)$ `// f is the Leaky ReLU activation`

for $l \leftarrow 2$ **to** L **do**

  $H_l \leftarrow f(W_l H_{l-1} + b_l)$

end

$pred\_labels \leftarrow softmax(H_L W_s + b_s)$

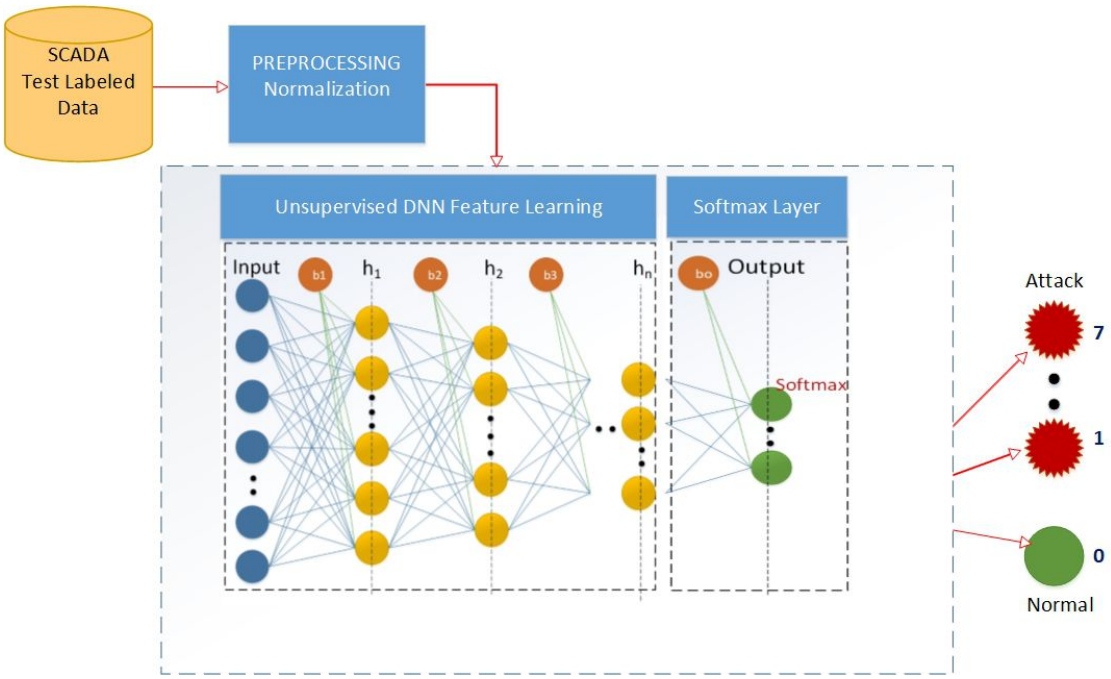

Figure 5.8 – DNN ADS Detection

5.7 Distributed Hybrid ADS for SCADA Networks

Distributed Machine Learning frameworks have been proposed by Google Research Team.

5.7.1 First Generation of Distributed Machine Learning Systems : DistBelief

The first generation of Distributed Machine Learning Systems was the DistBelief [39], a software framework that can use computer cluster with thousands of machines to train large models. On each node, the user defines the computation and the message that should be passed during the computation. In case of large models, the user can partition it across several machines of the cluster. DisBelief seamlessly manages the parallelization of the computation, the communication, the synchronisation as well as data transfer between machine. To enable the distribution of the training across multiple model instances, they propose two new large-scale distributed optimization procedures i.e. Downpour SGD and Sandblaster L-BFGS which are distributed versions of the Stochastic Gradient Descent (SGD) and the Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [171] method respectively.

In the Downpour SGD approach, the training dataset is divided into a number of subsets and a copy of the model is run on each of the subset. A centralized parameter server keeps the current state of all models parameters scattered across the nodes. On the other hand, the Sandblaster L-BFGS uses a different approach where a coordinator issues commands like dot product, scaling, coefficient-wise addition, multiplication, etc., that is performed by each parameter server shard independently, with the results being stored locally on the same shard.

5.7.2 Second Generation of Distributed Machine Learning Systems : TensorFlow

Thereafter, in 2015, the same Google Research team proposed the second generation and of Distributed Machine Learning Framework which is Tensorflow [2]. TensorFlow computation is a directed graph which is composed of a set of nodes (Figure 5.9).

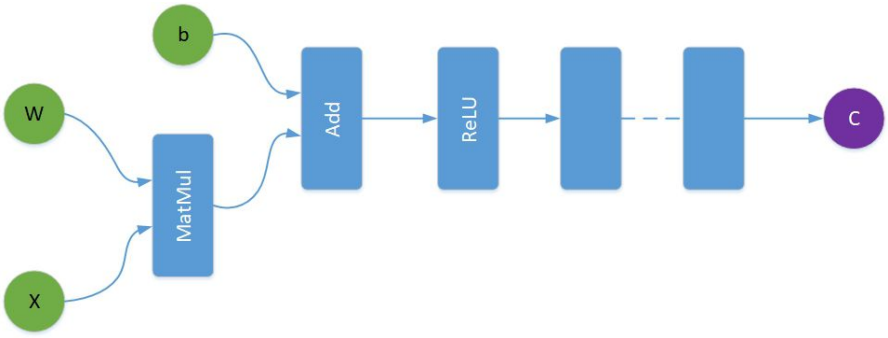

Figure 5.9 – TensorFlow Computation Graph

In TensorFlow, the client which is the main component uses Session interface to communicate with the master, and one or more worker processes. The graph nodes are executed on computational devices such as classical CPU or Graphical Processing Units (GPU). Each worker process manages one or more devices.

Two approaches exist in distributed TensorFlow :

Data Parallel Training

In this case, to speed up Stochastic Gradient Descent (SGD) training algorithm, we parallelize the computation of the gradient for a mini-batch across mini-batch elements i.e., the TensorFlow graph has many replicas of the portion of the graph doing the massive training, and a single client thread drives the entire training loop for this large graph.

Training and Model Parallel Training

In Model parallel training, different portions of the model computation are done on different computational devices simultaneously for the same batch of example.

5.7.3 Proposed Distributed Deep Neural Network Anomaly Detection System for SCADA

In our approach, multiple autoencoders are stacked, and the feature learning uses a greedy layer-wise pre-training. But it is shown in practice that in deep neural networks, going deeper than 3 layers does not help much more in terms of performance [74]. Our main focus is the size of data that could be very important in SCADA networks. The distributed approach of the hybrid deep neural network anomaly detection system for SCADA will then use the data parallelism to speed up the training process. The calculation of the gradient in the model training that consists of multiplication of large matrices while iterating over a large dataset is time consuming. To speed up the Stochastic Gradient Descent training algorithm, we parallelize the computation of the gradient for a mini-batch across mini-batch elements [2]. Several distributed deep learning like Dryad [77], Flume [25], CIEL [117], Naiad [118] and Spark [167] exist. But the Distributed TensorFlow framework [2] uses a hybrid data flow model that borrow elements from the previous frameworks without their shortcomings. Furthermore, TensorFlow is more flexible and allows the expression of a wide variety of machine leaning models and optimization algorithms compared to distributed deep learning approaches like DistBelief[39], the Adam Project [31] and the Parameter Server Project [36]. An Hadoop [160] cluster with different nodes will be used and the training data will be stored on the Hadoop HDFS database. Thus, the TensorFlow distributed framework will parallelize the computation of the gradient of the mini-batches across the different nodes and a parameter server with update the weights on the master (Figure 5.10).

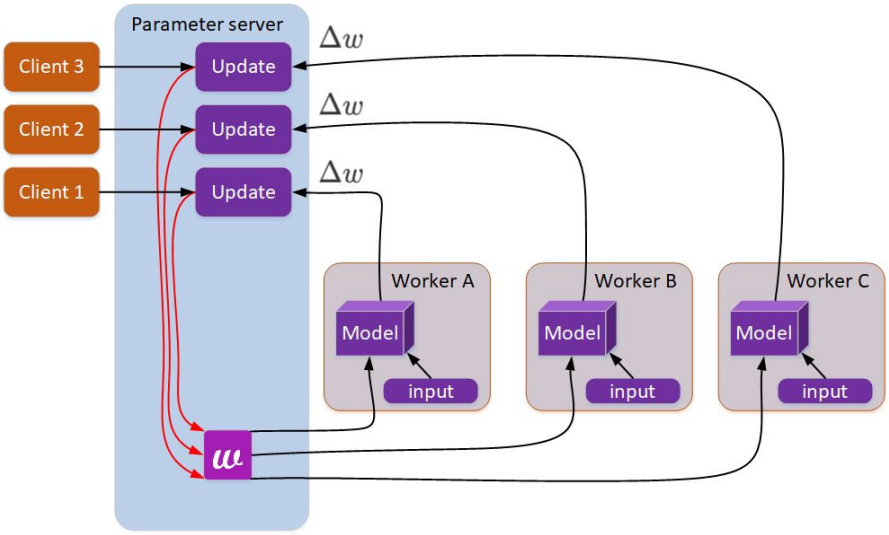

Figure 5.10 – Distributed DNN-based Anomaly Detection System for SCADA Networks Architecture

5.8 Conclusion

In this chapter, we provide the global architecture of the proposed anomaly detection system which has a pre-processing engine, and a hybrid deep neural network anomaly detection engine. The pre-processing consists in normalizing, balancing, one-hot encoding the labels, and splitting the data into training, test and validations sets.
The hybrid deep neural network anomaly detection engine is obtained by adding a softmax classification layer on the top of the unsupervised deep feature learner designed in the previous chapter. The cross-entropy is used as the loss function for the supervised training and the whole hybrid deep neural network training. The hybrid DNN anomaly detection system is trained in three steps i.e., a greedy layer-wise unsupervised pre-training step, a supervised softmax layer training step and the whole network fine-tuning step. The derived model from the training process is then used for the detection phase.

As the training time is a big challenge in deep learning approaches, we proposed a distributed approach of the hybrid DNN anomaly detection system which uses the distributed TensorFlow framework with a parameter server which stores and updates the model parameter, then distributes it to each worker node where the time consuming gradient calculation takes place.

In the next chapter, we do the implementation of the proposed approach on a single machine and a Hadoop cluster distributed environment, and discuss the different results.

Chapter 6

Implementation and Results

6.1 Introduction

This last chapter is intended for the implementation of the proposed hybrid deep neural network based anomaly detection systems for SCADA networks. After the introduction, the second section shows the development environment setup used for this thesis. The following section, gives details of the second dataset used i.e. the gas pipeline SCADA dataset. The fourth and fifth sections respectively illustrates the preparation of the two datasets before usage and their distributions. In the sixth section, we show the different performance measures used to compare the approach with baseline methods. Sections seven through nine are used to show the different results of the proposed approach with the SCADA water storage tank and gas pipeline datasets. We discuss about the experimentation results in section ten before a conclusion.

6.2 Development environment setup

For the different experiments of the present work, we setup an infrastructure via OpenStack. The hybrid deep neural network anomaly detection system was implemented on a server with 8vcpu, 8 Go of RAM and 10 Go hard drive. The server is running a Ubuntu 18.04.2 LTS operating system. The Deep Learning framework used is TensorFlow 1.5.0 running on Python 2.7.

For the distributed approach of the anomaly detection system, we setup an Hadoop cluster (Figure 6.1). The Hadoop version is 2.7.3. The cluster has one master and five workers. Each node has 8vcpu, 8 Go of RAM, 10 Go hard drive and runs a Ubuntu 18.04.2 LTS operating system. Distributed Deep Learning is powered by Distributed TensorFlow (version 1.5.0) running on python 2.7. The HDFS file system on the master node is used to store the SCADA datasets.

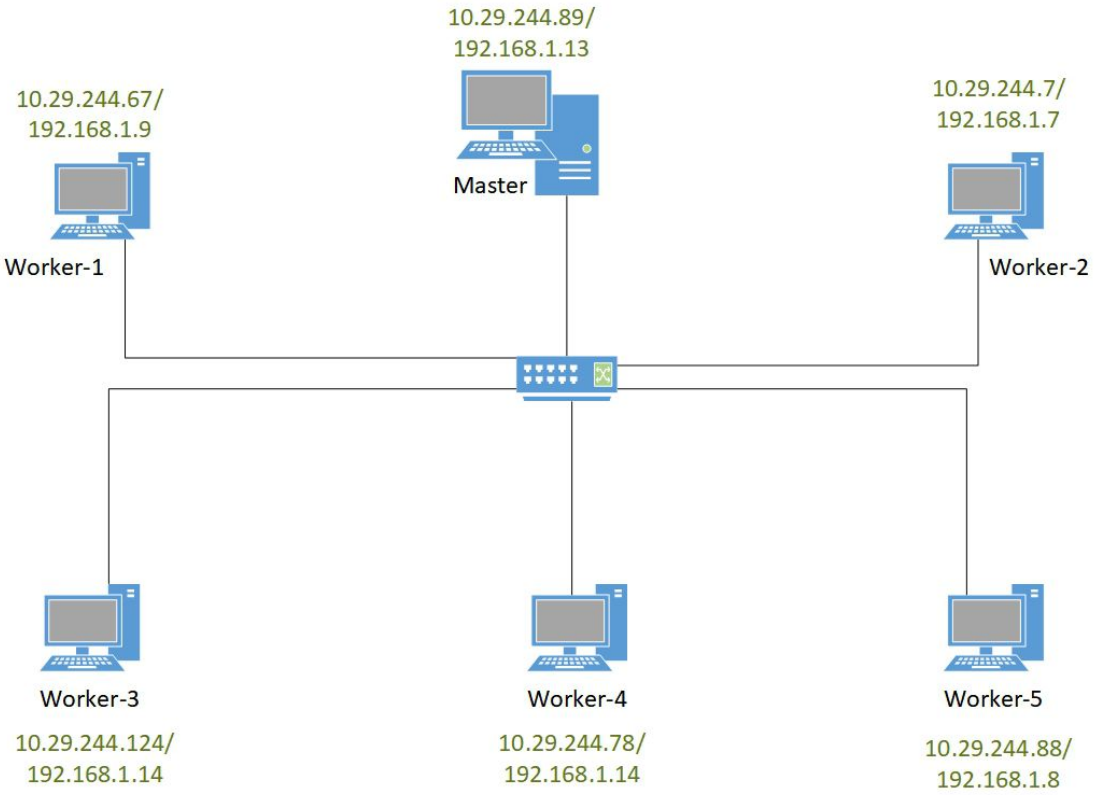

Figure 6.1 – Hadoop Cluster

6.3 SCADA Datasets used

6.3.1 Water Storage Tank SCADA dataset

In chapter 4, we gave a broad description of the Water Storage Tank System SCADA dataset used (Figures 6.2 and 6.3). Table 6.1 and Table 6.2 respectively show the water storage tank SCADA dataset records categories and attributes.

Figure 6.2 – Water Storage Tank
Source: [115]

6.3.2 Gas Pipeline SCADA dataset

The gas pipeline system as the water storage tank system is a laboratory-scale SCADA system from the Mississippi State University [115] . The gas pipeline system (Figures 6.4 and 6.5) includes an airtight pipeline connected to a compressor, a pressure meter and a solenoid-controlled relief valve. The pipeline system attempts to maintain the air pressure in the pipeline using a proportional integral derivative (PID) control scheme [115].
The gas pipeline system dataset contains the same normal and attacks records categories like in Table 6.1, previously described in chapter 4. But the payload attribute of both systems differ, though having the same network attributes (6.3).

Figure 6.3 – Water Storage Tank HMI
Source: [115]

Table 6.1 – Dataset records categories

Label	Category	Description
0	Normal	Instance not part of an attack
1	NMRI	Naive Malicious Response Injection attack
2	CMRI	Complex Malicious Response Injection attack
3	MSCI	Malicious State Command Injection attack
4	MPCI	Malicious Parameter Command Injection attack
5	MFCI	Malicious Function code Command Injection attack
6	DoS	Denial-of-Service attack
7	Reconnaissance	Reconnaissance attack

Table 6.2 – water storage tank attributes

Attribute	Type	Description
command_address	Network	Device ID in command packet
response_address	Network	Device ID in response packet
command_memory	Network	Memory start position in cmd.
response_memory	Network	Memory start position in resp.
command_memory_count	Network	Memory bytes for R/W command
response_memory_count	Network	Memory bytes for R/W response
comm_read_fun	Payload	Value command read func.code
comm_write_fun	Payload	Value command write func.code
response_read_fun	Payload	Value response read func. code
response_write_fun	Payload	Value response write func.code
sub_function	Payload	Value of sub-function code
command_length	Network	Total length of command packet
response_length	Network	Total length of response packet
HH	Payload	Value of HH setpoint
H	Payload	Value of H setpoint
L	Payload	Value of L setpoint
LL	Payload	Value of LL setpoint
control_mode	Payload	Automatic, manual or shutdown
control_scheme	Payload	Manual mode compressor/pump
pump_state	Payload	Compressor/pump state
crc_rate	Network	CRC error rate
measurement	Payload	water level
time	Network	Time interval betw. 2 packets
label	Provided	Manual classification

6.4 SCADA Datasets Preparation

The Water Storage Tank dataset and the Gas Pipeline dataset went through a preparation process i.e. a Min-Max normalization that sets all the datasets values except the labels in the range [0,1], the splitting of the datasets into a training dataset (60 %), a validation dataset (20 %) and test dataset (20 %) using the hold-out cross-validation [133], the balancing of training and validation datasets, and finally the one-hot encoding of all the datasets labels.

Figure 6.4 – Gas pipeline System
Source: [115]

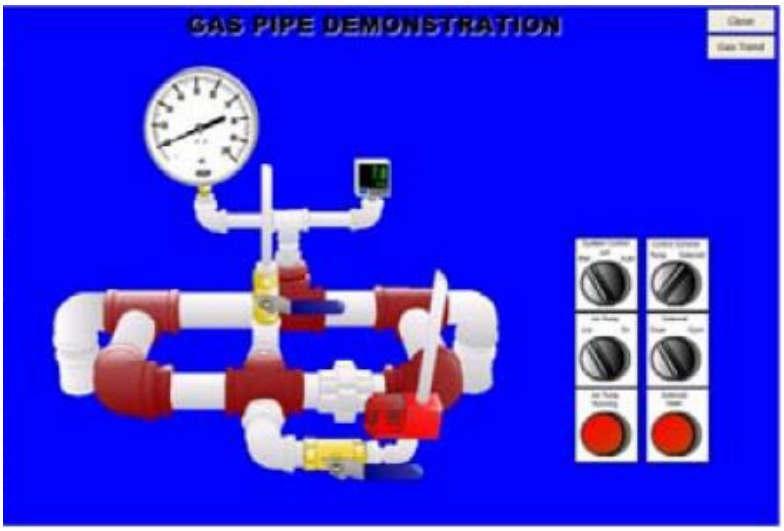

Figure 6.5 – Gas Pipeline System HMI
Source: [115]

6.5 Datasets distribution

It is a good practice to have a general idea of the class distribution of training and test datasets. As we are using the hold-out cross validation technique, we have partitioned the different datasets into training, validation and test dataset (60 %, 20 %, and 20 % respectively). The datasets contains eight categories

Table 6.3 – Gas pipeline attributes

Attribute	Type	Description
command_address	Network	Device ID in command packet
response_address	Network	Device ID in response packet
command_memory	Network	Memory start position in cmd.
response_memory	Network	Memory start position in resp.
command_memory_count	Network	Memory bytes for R/W command
response_memory_count	Network	Memory bytes for R/W response
comm_read_fun	Payload	Value command read func.code
comm_write_fun	Payload	Value command write func.code
response_read_fun	Payload	Value response read func. code
response_write_fun	Payload	Value response write func.code
sub_function	Payload	Value of sub-function code
command_length	Network	Total length of command packet
response_length	Network	Total length of response packet
Set_point	Payload	Target gas pressure in the pipe
solenoid_state	Payload	State of solenoid used to open the gas relief valve
gain	Payload	Gain parameter value of the PID controller
reset	Payload	Reset parameter value of the PID controller
dead_band	Payload	Dead band parameter value of the PID controller
cycletime	Payload	Cycle time parameter value of the PID controller
control_mode	Payload	Automatic, manual or shutdown
control_scheme	Payload	Manual mode compressor/pump
pump_state	Payload	Compressor/pump state
crc_rate	Network	CRC error rate
measurement	Payload	water level
time	Network	Time interval betw. 2 packets
label	Provided	Manual classification

of records i.e. one category of normal records (class 0) and seven categories of attacks (classes 1 through 7). We also considered a version of the datasets containing two categories of records i.e., one category of normal records (class 0) and one category of anomalous records (class 0). For the anomalous, records, we have binarized the dataset i.e., all the attacks records are labeled 1. Figures 6.4 through 6.13 represent the data distribution of the water storage tank and gas pipeline SCADA training and test dataset (8-class and binary). We notice that the datasets are very imbalanced, e.g., the water storage tank training dataset

normal records times more than 140 the attacks of category 6 (DoS) records, or the gas pipeline normal records is 110 higher than the class 5 (MFCI) attack records. The test sets are imbalanced as well. To avoid undesirable bias in the training process, the training data subsets undergo a balancing process before usage, but all the test data subsets are kept as is.

Table 6.4 – Water Storage Tank dataset distribution eight-class

	Total	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Full dataset	236179	172415	9187	12460	1833	3725	1320	1237	34002
Training	141707	103489	5539	7532	1073	2265	982	727	20300
Validation	47236	33586	2033	2785	414	690	153	271	7304
Test	47236	34463	1824	2464	380	730	169	255	6851

Table 6.5 – Water Storage Tank dataset distribution two-class

	Total	**0**	**1**
Full dataset	236179	172415	63764
Training	141707	103489	38182
Validation	47236	33586	13650
Test	47236	34463	12791

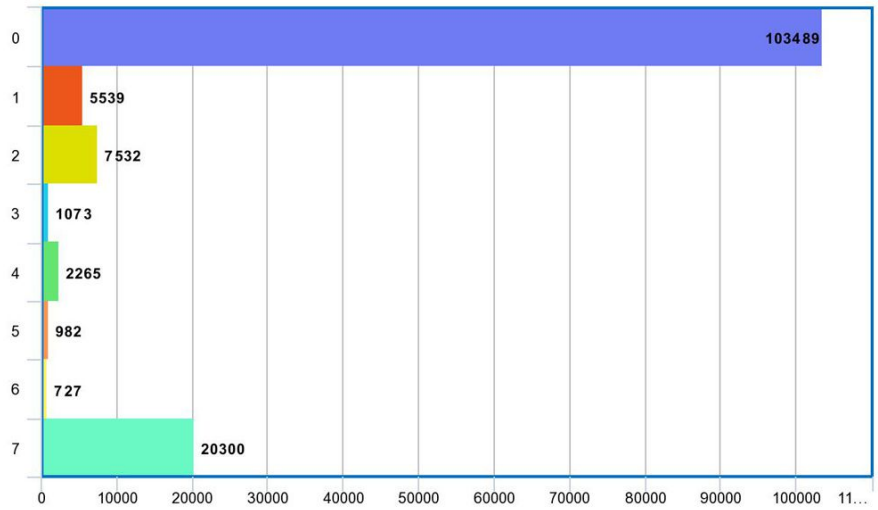

Figure 6.6 – Water Storage Tank Training Data distribution (eight-class)

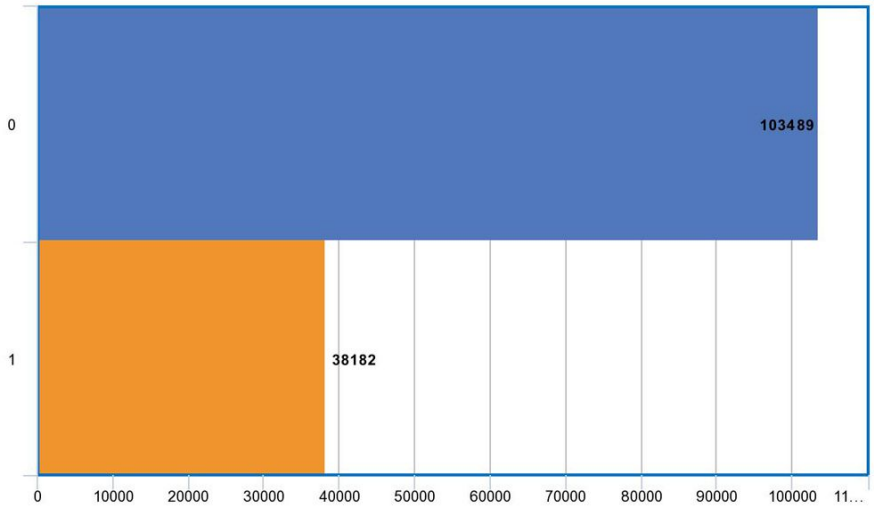

Figure 6.7 – Water Storage Tank Training Data distribution (two-class)

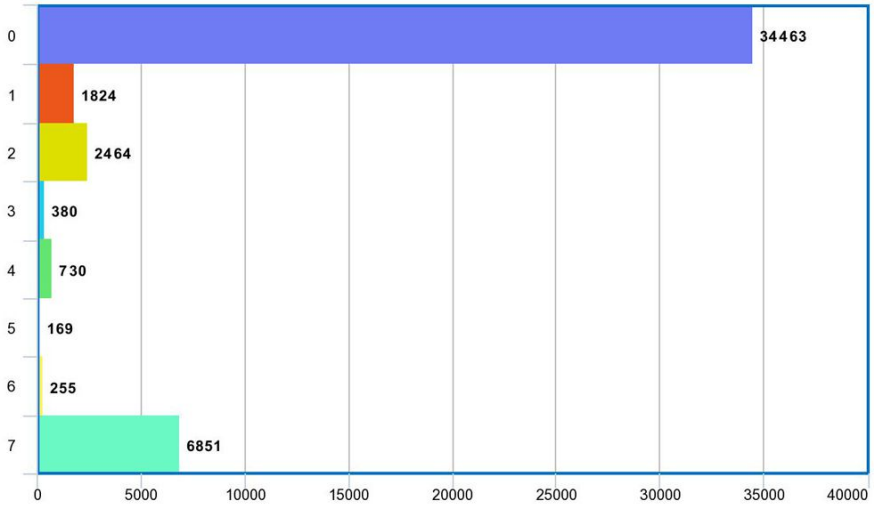

Figure 6.8 – Water Storage Tank Test Data distribution (eight-class)

Table 6.6 – Gas Pipeline dataset distribution eight-class

	Total	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Full dataset	97019	61156	2763	15466	782	7637	573	1837	6805
Training	58211	36502	1641	9240	482	4637	331	1109	4269
Validation	19404	13408	603	2546	175	1202	147	298	1025
Test	19404	12327	561	3113	150	1500	121	364	1268

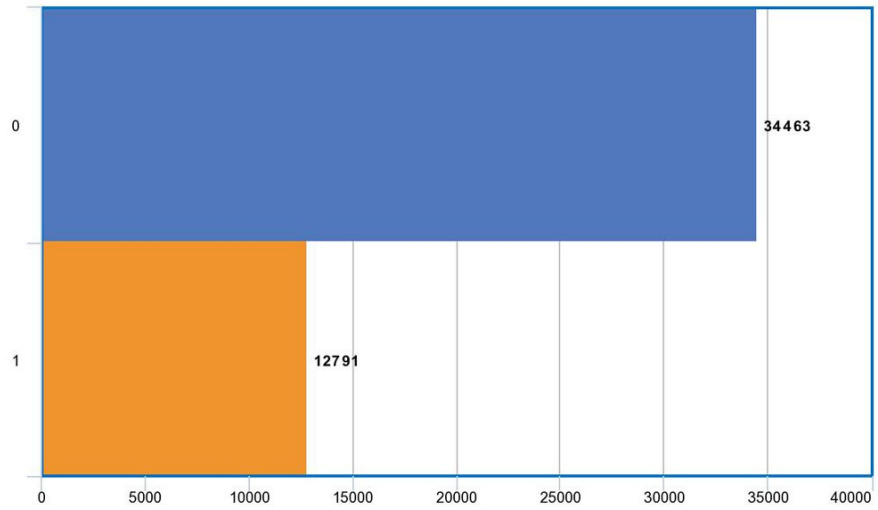

Figure 6.9 – Water Storage Tank Test Data distribution (two-class)

Table 6.7 – Gas Pipeline dataset distribution two-class

	Total	**0**	**1**
Full dataset	97019	61156	35863
Training	58211	36502	21709
Validation	19404	13408	5996
Test	19404	12327	7077

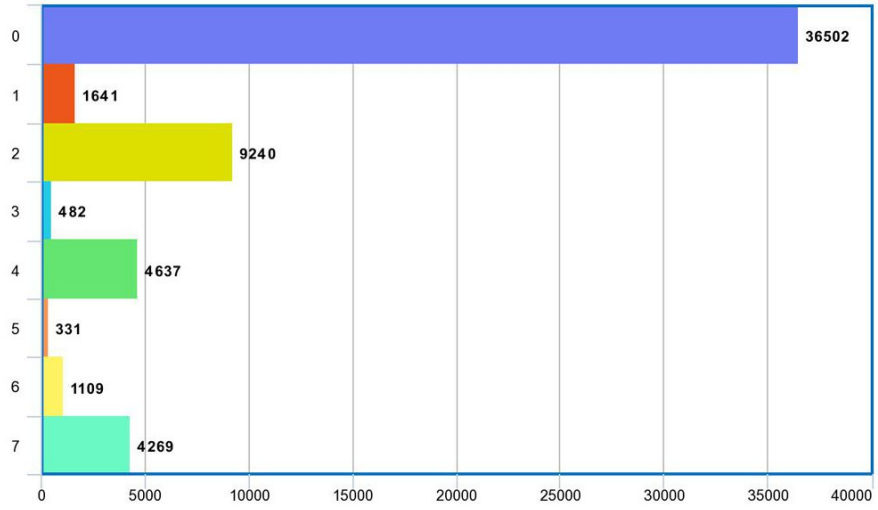

Figure 6.10 – Gas Pipeline Training Data distribution (eight-class breakdown)

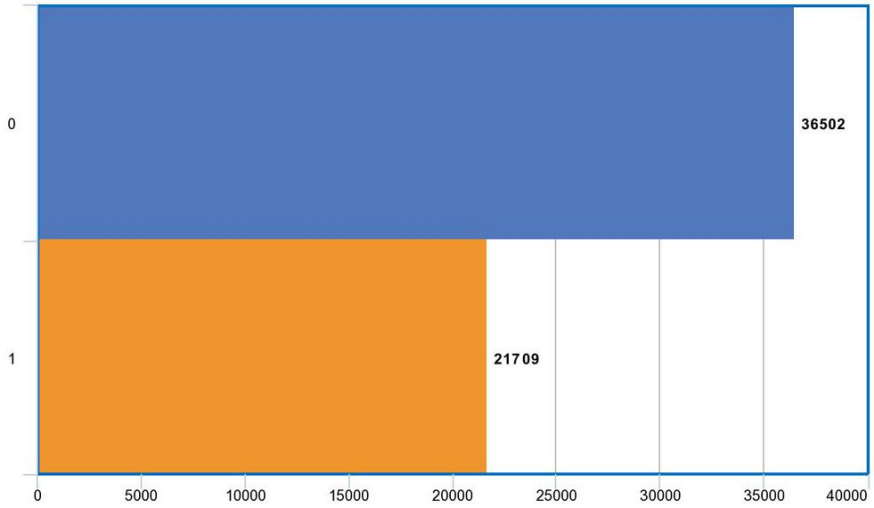

Figure 6.11 – Gas Pipeline Training Data distribution (two-class breakdown)

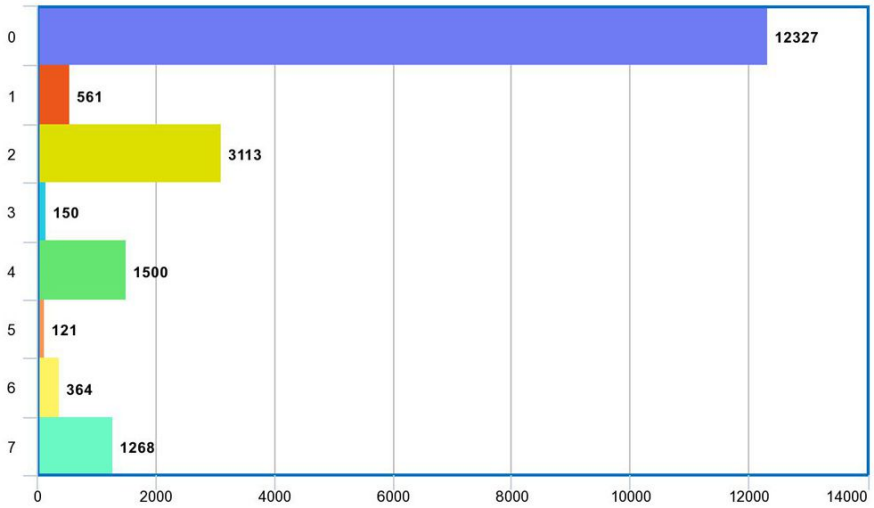

Figure 6.12 – Gas Pipeline Test Data distribution (eight-class breakdown)

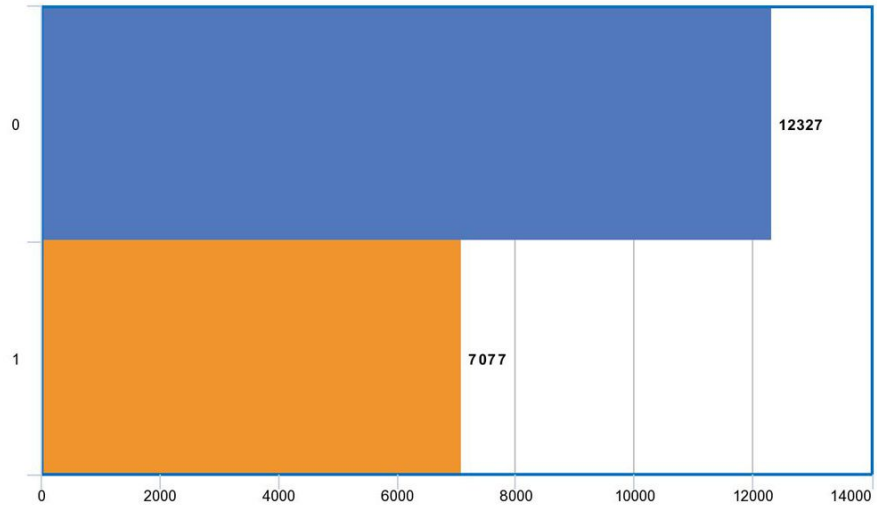

Figure 6.13 – Gas Pipeline Test Data distribution (two-class breakdown)

6.6 Performance measures

Accuracy, recall (or sensitivity), precision, specificity, f1-score ¨[63], along with training time and prediction time , [151] are the measures used to evaluate the performances of classifiers. *Positive tuples are the tuples of main interest* in a dataset.

In the present work, the anomaly detection system is detecting anomalies. Hence, the positive tuples are the abnormal ones.

Negative tuples are all the other tuples.

Let P be the number of positive tuples and N the number of negative tuples.

True Positive (TP) represents positive tuples that were correctly labeled by the classifier.
True Negative (TN) represents negative tuples that were correctly labeled by the classifier.
False Positive (FP) represents negative tuples that were incorrectly labeled as positive.
False Negative (FN) represents positives tuples that were incorrectly labeled as negative.

Accuracy (or recognition rate)

$$accuracy = \frac{TP + TN}{P + N}$$

For imbalanced datasets (i.e main class of interest rare), the Accuracy measure is not really relevant. Other metrics such as recall, precision and f1-score are used instead.

recall
The recall is the True Positive rate. It gives the proportion of positive tuples

correctly identified.

$$recall = \frac{TP}{P}$$

Precision

Precision is a measure of exactness. It answers to the question *What percentage of tuples labeled as positive are actually such?*

$$precision = \frac{TP}{TP + FP}$$

False Positive Rate (FPR)

The False Positive Rate (FPR) or False Alarm Rate (FAR) is the proportion of falsely identifying a normal tuple as anomalous.

$$FPR = \frac{FP}{FP + TN}$$

False Negative Rate (FNR)

The False Negative Rate (FNR) is the proportion of falsely identifying an anomalous tuple as a normal one.

$$FNR = \frac{FN}{FN + TP}$$

f1-score

Precision is a measure of harmonic mean.

$$f1 - score = \frac{2 * (recall * precision)}{recall + precision}$$

6.7 Results for Water Storage Tank Dataset

6.7.1 Models comparison

The design process of neural networks i.e. defining the parameters and the hyper-parameters consists of trial and error, guessing what king of hidden unit, the width, the depth of the network or which learning rate works well [59] [128]. The network will then be trained, and the parameters and hyper-parameters are evaluated with a validation set.
We have conducted various tests on the datasets by varying the architecture of the stacked autoencoders to 2, 3, 4 and 5 layers, with various parameters and hyperparameters configuration. Specifically, we did the tests for different training epochs i.e. 1, 10, 20, 30, 40, 50.

The Table 6.8 shows two different architectures used for comparison sake, and their training results in Table 6.9 and 6.10.

Table 6.8 – Two Hybrid DNN Architectures

	Architecture 1	**Architecture 2**	**Common parameters /hyper-parameters**
Nb. of hidden layers	2	3	Activation function : Leaky ReLU Loss function : Cross-Entropy Learning rate : 0.01 Sparsity : 0.05 Noise level : 30 %
Nb. input nodes	24	24	
Nb. of nodes per layer	$L1 = 52$ $L2 = 34$	$L1 = 30$ $L2 = 27$ $L3 = 40$	
Nb. Output nodes	8	8	

Table 6.9 – Loss, Accuracy, Training Time Per Epochs for Water Storage Tank Dataset 3 layers

Epochs	**1**	**10**	**20**	**30**	**40**	**50**
Training Loss	260.16	161.76	143.20	142.54	139.04	138.57
Validation Loss	248.10	148.32	139.06	142.96	148.70	152.36
Accuracy	89 %	88 %	89 %	90 %	90 %	90 %
Single Machine Training Time (s)	32.532	141.890	280.686	418.449	554.043	693.971
Distributed Cluster Training Time (s)	8.025	29.374	60.133	87.682	120.095	155.708

Table 6.10 – Loss, Accuracy, Training Time Per Epochs for Water Storage Tank Dataset 2 layers

Epochs	**1**	**10**	**20**	**30**	**40**	**50**
Training Loss	287.15	190.02	165.56	152.89	144.99	142.11
Validation Loss	260.33	169.45	152.03	145.22	144.17	151.25
Accuracy	85 %	86 %	86 %	88 %	88 %	88 %
Single Machine Training Time (s)	25.224	118.111	244.505	365.201	480.996	602.306
Distributed Cluster Training Time (s)	6.005	23.561	44.774	67.895	95.665	127.354

Figures 6.9 and 6.10 show the loss curves of the two architectures while 6.16 and 6.17 show their respective accuracy curves.

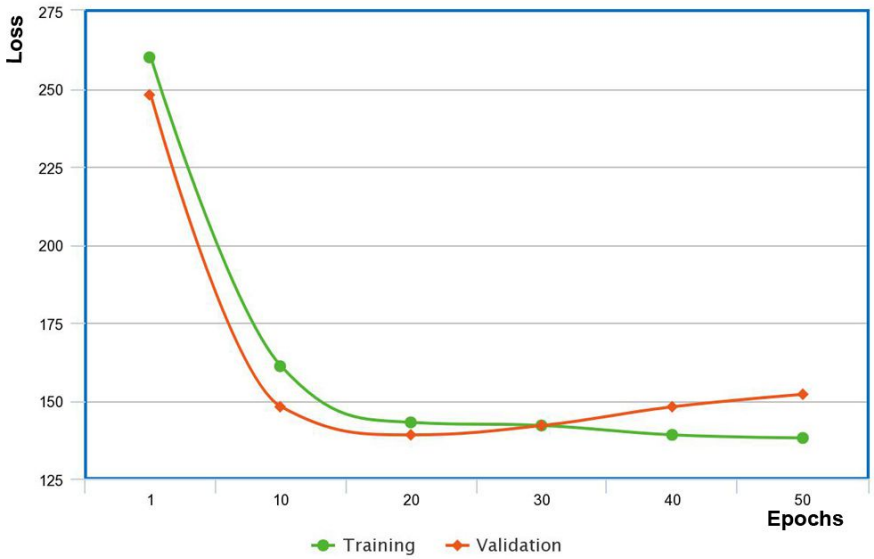

Figure 6.14 – Water Storage Tank Loss Curve 3 hidden layers

The 2 hidden layer architecture loss curve shows that while training loss continue to decrease after 40 epochs, validation loss increases. So the best results are obtained after 40 epochs of training. Likewise, the best results are obtained after 30 epochs of training for the 3 hidden layer architecture.
The two accuracy curves show a better accuracy for the 3 hidden layer architecture.

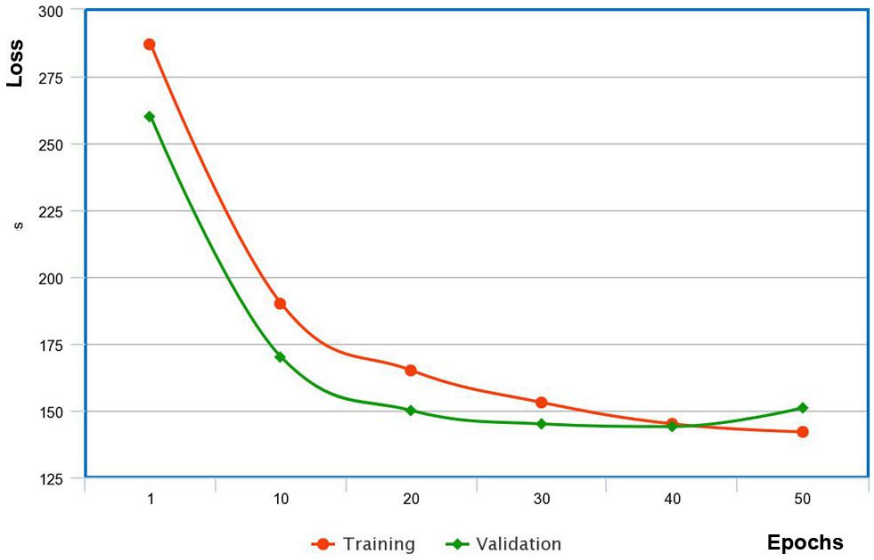

Figure 6.15 – Water Storage Tank Loss Curve 2 hidden layers

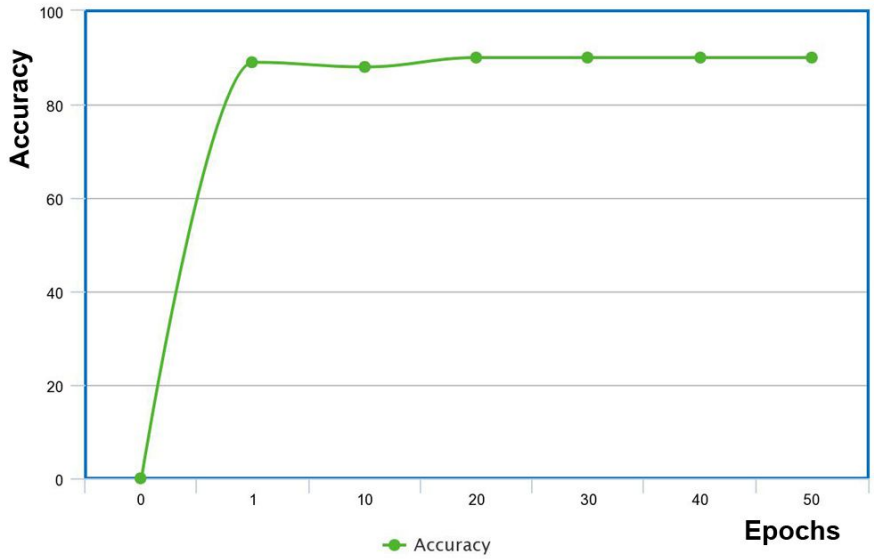

Figure 6.16 – Water Storage Tank Accuracy Curve 3 hidden layers

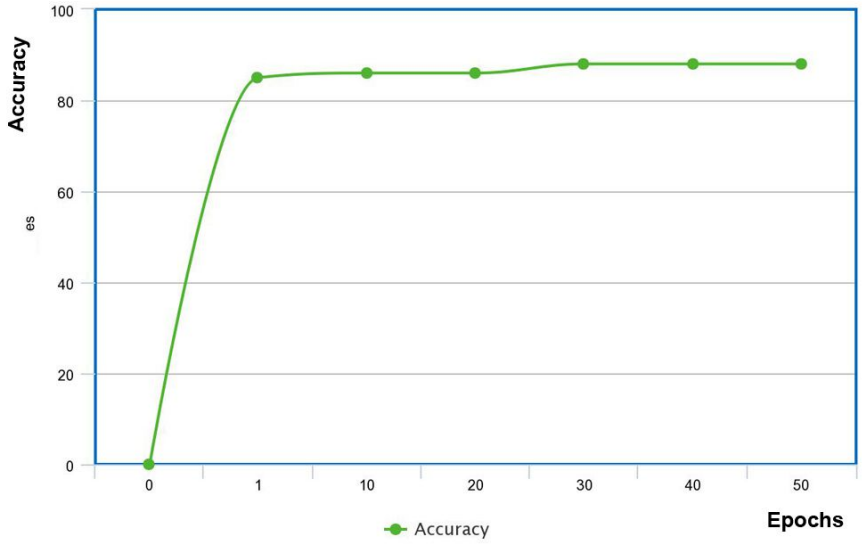

Figure 6.17 – Water Storage Tank Accuracy Curve 2 hidden layers

6.7.2 Experimental results for the water storage tank dataset

We give here the best results obtained for the SCADA Water Storage Tank dataset. The results using the multiclass version of the water storage tank SCADA dataset were obtained for 30 epochs of training, an architecture with 3 hidden layers, a Leaky ReLU activation function, a cross-entropy loss function, a batch size of 128, a learning rate of 0.01 a sparsity parameter of 0.05, a noise level of 30%, and 30, 27, 40 number of nodes for layer 1, layer 2 and layer 3 respectively. After 30 epochs, as shown on Figure 6.15, the validation loss increases even though the training loss continue to decrease. This is why we choose the results at 30 epochs. On the other hand, the results for the binary Water Storage Tank SCADA dataset is a model trained during 30 epochs, which has 3 hidden layers, a batch size of 128, a learning rate of 0.01 a sparsity parameter of 0.05, a noise level of 30%, and 38, 24, 20 number of nodes for layer 1, layer 2 and layer 3 respectively.

For the multiclass classification with the Water Storage Tank SCADA dataset (Table 6.19), we have a detection rate of 0 % for attack classes 1 and 2. Attack class 6 also have a detection rate of only 35 %. The 36 % of False Positive Rate for normal records detection confirms the previous results, as class 1, 2 and 6 as mis-classified as normal data. The binary classification for the same dataset (Table 6.12) gives a 66 % detection rate, due to a high False Negative rate (34 %). This clearly shows that some of the attacks are not detected, thus classified as normal records.

Table 6.11 – Hybrid DNN ADS Water Storage Tank Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	90 %	100 %	88 %	36 %	0 %	94 %	34418
1 (NMRI)	96 %	0 %	0 %	0 %	100 %	0 %	1848
2 (CMRI)	95 %	0 %	0 %	0 %	100 %	0 %	2438
3 (MSCI)	100 %	92 %	97 %	0 %	8 %	94 %	380
4 (MPCI)	100 %	87 %	99 %	0 %	13 %	93 %	742
5 (MFCI)	100 %	85 %	100 %	0 %	15 %	92 %	259
6 (DoS)	100 %	35 %	99 %	0 %	65 %	51 %	225
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	6925

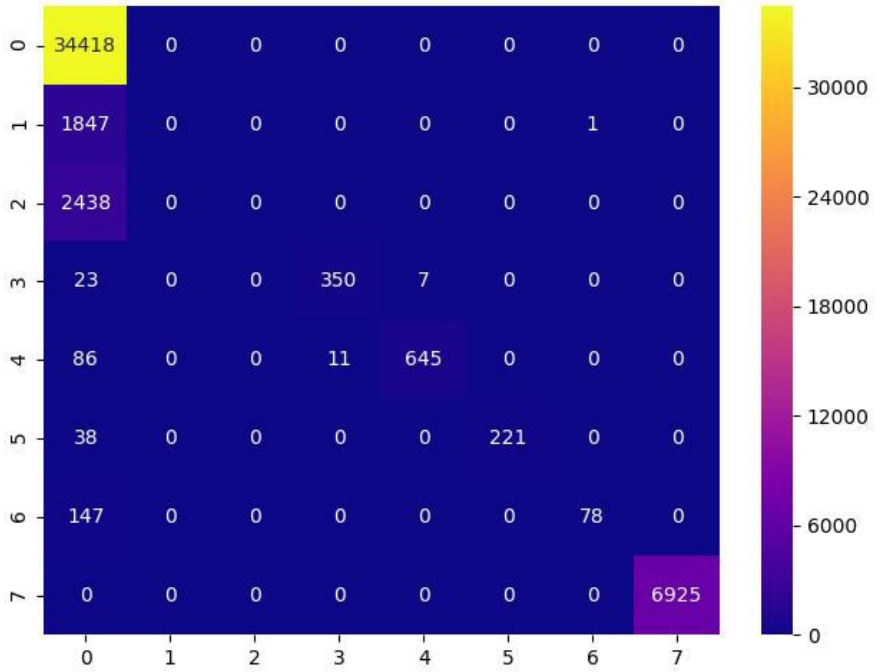

Figure 6.18 – Confusion Matrix Water Storage Tank Dataset Multiclass Classification

Table 6.12 – Water Storage Tank Binary Classification

	Accuracy	**Recall**	**Precision**	**FPR**	**FNR**	**F1-score**
Hybrid DNN	91 %	66 %	100 %	0 %	34 %	79 %

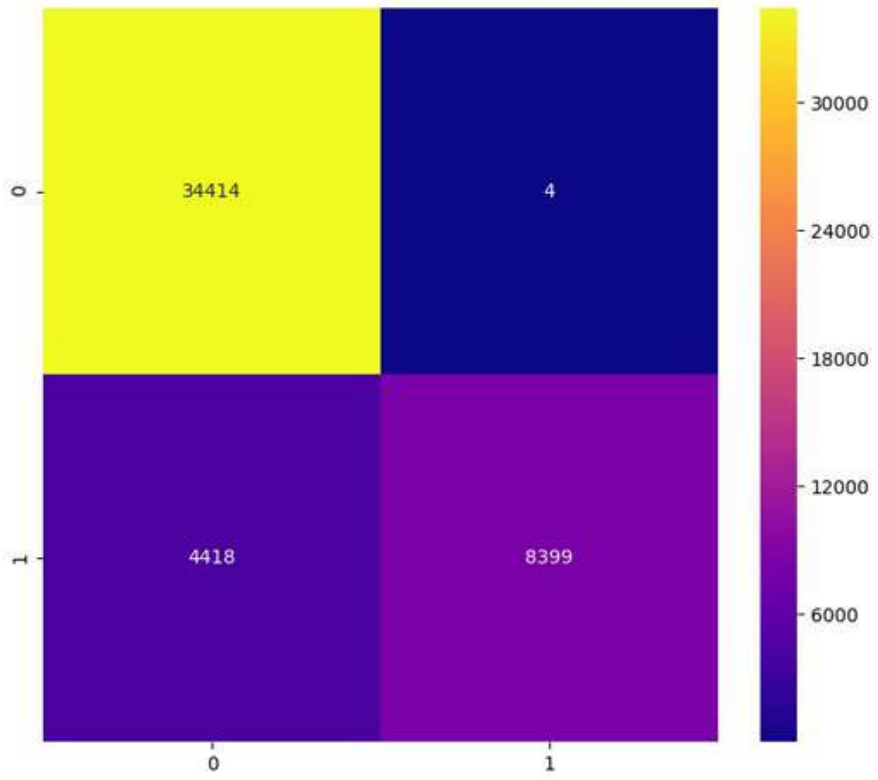

Figure 6.19 – Confusion Matrix Water Storage Tank Dataset Binary Classification

6.7.3 Single vs Distributed Hybrid DNN SCADA Anomaly Detection Results

For the distributed approach, we create the cluster which has a parameter server and the worker nodes, then we specify the master as the parameter server and the training job is assigned to the workers. Each worker takes care of only a task which is a portion of the training job.
Finally, this time, the SCADA data are read from the Hadoop HDFS database.

The command lines used to run the distributed model is :

```
#### Command line parameter server #####
$python dist_sae.py --ps_hosts=192.168.1.13:2222
--worker_hosts=192.168.1.9:2222,192.168.1.7:2222,
192.168.1.14:2222,192.168.1.22:2222,192.168.1.8:2222
--job_name=ps --task_index=0

#### Command line for each worker #####
$python dist_sae.py --ps_hosts=192.168.1.13:2222
--worker_hosts=192.168.1.9:2222,192.168.1.7:2222,
192.168.1.14:2222,192.168.1.22:2222,192.168.1.8:2222
--job_name=worker --task_index=0

$python dist_sae.py --ps_hosts=192.168.1.13:2222
--worker_hosts=192.168.1.9:2222,192.168.1.7:2222,
192.168.1.14:2222,192.168.1.22:2222,192.168.1.8:2222
--job_name=worker --task_index=1

$python dist_sae.py --ps_hosts=192.168.1.13:2222
--worker_hosts=192.168.1.9:2222,192.168.1.7:2222,
192.168.1.14:2222,192.168.1.22:2222,192.168.1.8:2222
--job_name=worker --task_index=2

$python dist_sae.py --ps_hosts=192.168.1.13:2222
--worker_hosts=192.168.1.9:2222,192.168.1.7:2222,
```

```
192.168.1.14:2222,192.168.1.22:2222,192.168.1.8:2222
--job_name=worker --task_index=3

$python dist_sae.py --ps_hosts=192.168.1.13:2222
--worker_hosts=192.168.1.9:2222,192.168.1.7:2222,
192.168.1.14:2222,192.168.1.22:2222,192.168.1.8:2222
--job_name=worker --task_index=4
```

The first command launches the program on the master which is the parameter server which stores the model and distributes it to the 5 workers. The master as a parameter server is also responsible for the model update. The five workers are charged of the bulk of the workload i.e. the computation of the gradient during the Back-propagation optimization algorithm. The computed gradient is sent by each worker to the parameter server for updating the model. The first worker with the task index 0 is the coordinator of the model training job done by the workers.

With the best model of the water storage tank SCADA dataset, we run the distributed version with that model setting in order to evaluate the training time for different epochs.
Figure 6.20 represents the training time of the 3 hidden-layer model of the hybrid DNN anomaly detection for the Water Storage Tank SCADA dataset for multiclass classification, on a single machine or a distributed cluster.

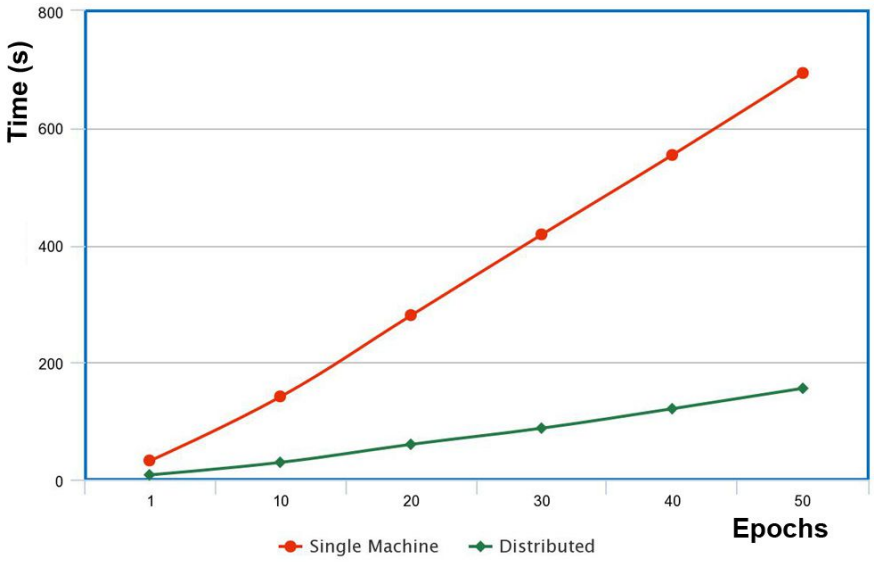

Figure 6.20 – Water Storage Tank Local vs Distributed Training Time

6.8 Results for Gas Pipeline Dataset

We did the test in the same conditions as previously i.e., using stacked autoencoders of 2, 3, 4 and 5 layers and 1, 10, 20, 30, 40, 50 epochs for each layer. The results are the best ones among all the various configurations for the multiclass and two class classification.
The results using the multiclass version of the Gas pipeline SCADA dataset were obtained after training the model with 40 epochs. After 40 epochs, as show on Figure 6.21, the validation loss increases even though the training loss continue to decrease. This motivates the choice of the results at 40 epochs. The model architecture has 2 hidden layers, a batch size of 128, a learning rate of 0.01 a sparsity parameter of 0.05, a noise level of 35%, and 58, 34 number of nodes for layer 1 and layer 2 respectively. On The other hand, the results for the binary water storage tank SCADA dataset architecture has 2 hidden layers, a batch size of 128, a learning rate of 0.01 a sparsity parameter of 0.05, a noise level of 35%, and 45, 30 number of nodes for layer 1 and layer 2 respectively.

Table 6.13 – Loss, Accuracy, Training Time Per Epochs for Gas Pipeline Dataset

Epochs	**1**	**10**	**20**	**30**	**40**	**50**
Training Loss	47.48	36.80	35.63	35.14	33.11	34.29
Validation Loss	47.20	36.99	34.01	33.087	33.78	36.13
Accuracy	95 %	95 %	95 %	95 %	95 %	95 %
Single Machine Training Time (s)	12.034	38.051	69.544	102.968	132.475	165.929
Distributed Cluster Training Time (s)	3.816	8.125	14.654	22.133	28.504	35.187

Figure 6.21 and 6.22 respectively represent the loss curve and accuracy curve of the Gas Pipeline SCADA dataset for multiclass classification.

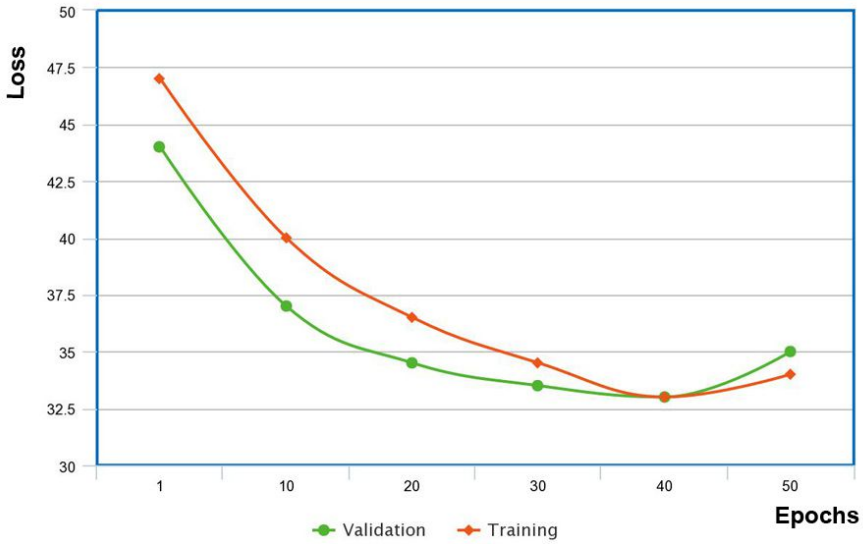

Figure 6.21 – Gas Pipeline Loss Curve 2 Layers

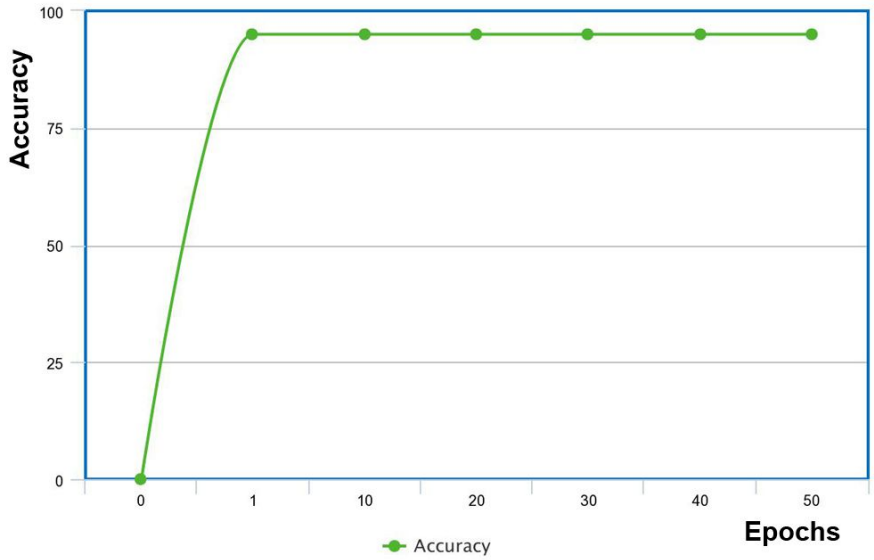

Figure 6.22 – Gas Pipeline Accuracy Curve 2 Layers

6.8.1 Experimental Results with the Gas Pipeline SCADA dataset

With the Gas Pipeline SCADA dataset, the results are better compared to the Water Storage Tank SCADA dataset, as only class 1 with a detection rate of 0 % is totally mis-classified. Here, the detection rate of class 6 improved to 76 %, and for the other class of attacks, the detection rate is above 94 %. The False Positive Rate drops to 10 % for normal records detection. Those good results are confirmed by binary classification with the Gas Pipeline SCADA dataset where we have a detection rate of 89 % and a low False Positive Rate of 2 %. The False Negative rate here has dropped to 11 %, meaning that the mis-classification of attacks as normal records is less important in this dataset.

Table 6.14 – Hybrid DNN ADS Gas Pipeline Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	95 %	98 %	94 %	10 %	2 %	96 %	12239
1 (NMRI)	97 %	0 %	0 %	0 %	100 %	0 %	541
2 (CMRI)	99 %	98 %	93 %	1 %	2 %	96 %	3103
3 (MSCI)	100 %	95 %	96 %	0 %	5 %	95 %	167
4 (MPCI)	100 %	98 %	97 %	0 %	2 %	98 %	1473
5 (MFCI)	100 %	94 %	94 %	0 %	6 %	94 %	124
6 (DoS)	100 %	76 %	100 %	0 %	24 %	87 %	391
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	1365

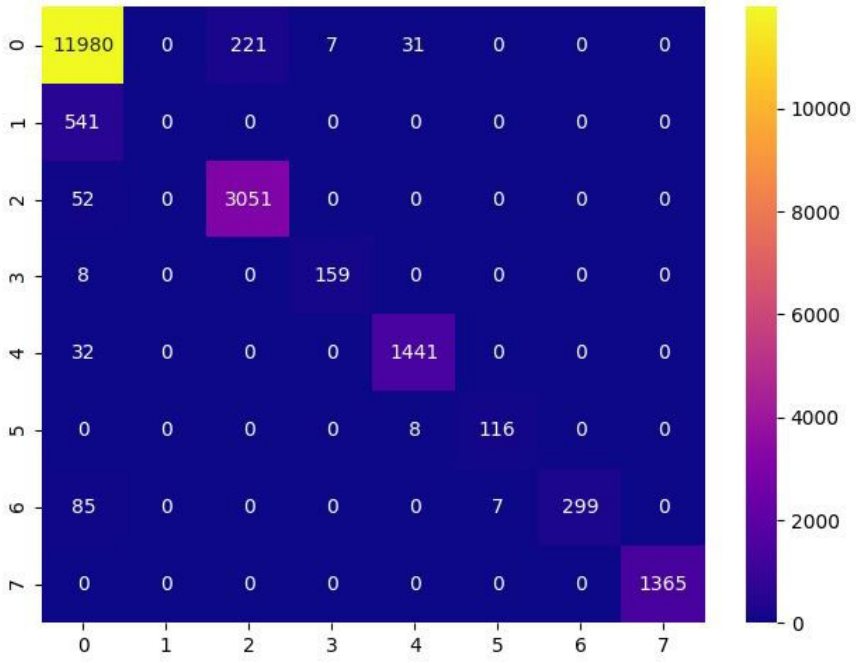

Figure 6.23 – Confusion Matrix Gas Pipeline Dataset Multiclass Classification

Table 6.15 – Gas Pipeline Binary Classification

	Accuracy	**Recall**	**Precision**	**FPR**	**FNR**	**F1-score**
Hybrid DNN	95 %	89 %	96 %	2 %	11 %	92 %

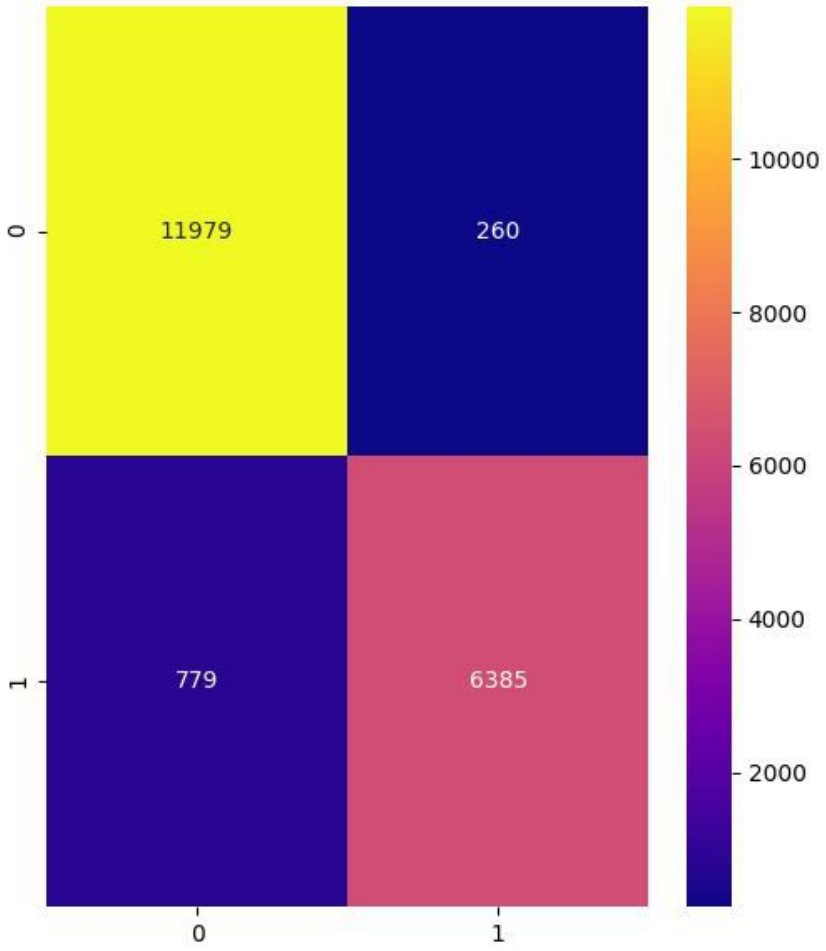

Figure 6.24 – Confusion Matrix Gas Pipeline Dataset Binary Classification

6.8.2 Single vs Distributed Hybrid DNN SCADA Anomaly Detection Results

Figure 6.25 represents the training time of the 2 layer model of the hybrid DNN anomaly detection for the Gas Pipeline SCADA dataset for multiclass classification, on a single machine or a distributed cluster.

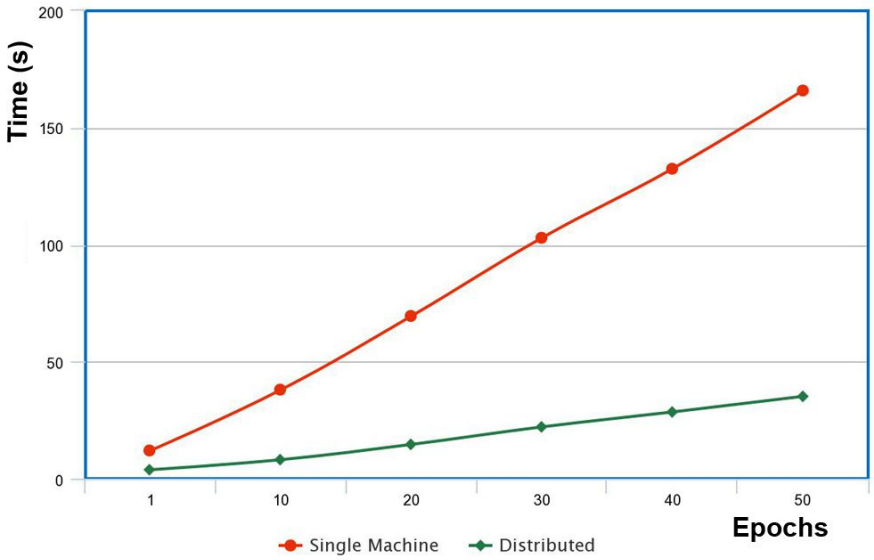

Figure 6.25 – Gas Pipeline Local vs Distributed Training Time

6.9 Results for Standard Approaches

The following tables represent the results obtained with the proposed approach as well as with the standards algorithms, i.e., Decision Tree, Naïve Bayes and Random Forest. The accuracy, recall, precision, False Positive Rates (FPR), False Negative Rates (FNR) and the F1-score performance measures are used to this end.

6.9.1 Water Storage Tank Dataset

Table 6.16 – Decision Tree Water Storage Tank Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	90 %	100 %	88 %	38 %	0 %	93 %	34418
1 (NMRI)	96 %	0 %	0 %	0 %	100 %	0 %	1848
2 (CMRI)	95%	0 %	0 %	0 %	100 %	0 %	2438
3 (MSCI)	99 %	0 %	0 %	0 %	100 %	0 %	380
4 (MPCI)	99 %	99 %	67 %	1 %	1 %	80 %	742
5 (MFCI)	99 %	0 %	0 %	0 %	100 %	0 %	259
6 (DoS)	100 %	0 %	0 %	0 %	100%	0 %	225
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	6925

Table 6.17 – Naïve Bayes Water Storage Tank Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	45 %	23 %	100 %	0 %	77 %	37 %	34418
1 (NMRI)	96 %	0 %	0 %	0 %	100 %	0 %	1848
2 (CMRI)	55%	100 %	11 %	56 %	0 %	19 %	2438
3 (MSCI)	99 %	96 %	58 %	1 %	4 %	72 %	380
4 (MPCI)	100 %	98 %	98 %	2 %	2 %	98 %	742
5 (MFCI)	100 %	100 %	100 %	1 %	0 %	100 %	259
6 (DoS)	100 %	100 %	3 %	18 %	0%	6 %	225
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	6925

Table 6.18 – Random Forest Water Storage Tank Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	91 %	100 %	89 %	34 %	0 %	94 %	34418
1 (NMRI)	96 %	0 %	0 %	0 %	100 %	0 %	1848
2 (CMRI)	95%	0 %	0 %	0 %	100 %	0 %	2438
3 (MSCI)	100 %	96 %	96 %	0 %	4 %	96 %	380
4 (MPCI)	100 %	98 %	99 %	0 %	2 %	98 %	742
5 (MFCI)	100 %	100 %	100 %	0 %	0 %	100 %	259
6 (DoS)	100 %	100 %	100 %	0 %	0%	100 %	225
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	6925

Table 6.19 – Hybrid DNN ADS Water Storage Tank Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	90 %	100 %	88 %	36 %	0 %	94 %	34418
1 (NMRI)	96 %	0 %	0 %	0 %	100 %	0 %	1848
2 (CMRI)	95 %	0 %	0 %	0 %	100 %	0 %	2438
3 (MSCI)	100 %	92 %	97 %	0 %	8 %	94 %	380
4 (MPCI)	100 %	87 %	99 %	0 %	13 %	93 %	742
5 (MFCI)	100 %	85 %	100 %	0 %	15 %	92 %	259
6 (DoS)	100 %	35 %	99 %	0 %	65 %	51 %	225
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	6925

Table 6.20 – Water Storage Tank Binary Classification

Approach	Accuracy	Recall	Precision	FPR	FNR	F1-score
Decision Tree	90 %	62 %	100 %	0 %	38 %	76 %
Naïve Bayes	91 %	65 %	100 %	0 %	35 %	79 %
Random Forest	91 %	66 %	100 %	0 %	34 %	79 %
Hybrid DNN	91 %	66 %	100 %	0 %	34 %	79 %

6.9.2 Gas Pipeline Dataset

Table 6.21 – Decision Tree Gas Pipeline Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	78 %	100 %	74 %	61 %	0 %	85 %	12239
1 (NMRI)	97 %	0 %	0 %	0 %	100 %	0 %	541
2 (CMRI)	84%	0 %	0 %	0 %	100 %	0 %	3103
3 (MSCI)	99 %	0 %	0 %	0 %	100 %	0 %	167
4 (MPCI)	100 %	98 %	98 %	0 %	2 %	98 %	1473
5 (MFCI)	99 %	0 %	0 %	0 %	100 %	0 %	124
6 (DoS)	98 %	0 %	0 %	0 %	100%	0 %	391
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	1365

Table 6.22 – Naïve Bayes Gas Pipeline Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	37 %	0 %	50 %	0 %	100 %	0 %	12239
1 (NMRI)	53 %	100 %	6 %	48 %	0 %	11 %	541
2 (CMRI)	88%	100 %	58 %	14 %	0 %	73 %	3103
3 (MSCI)	97 %	97 %	22 %	3 %	3 %	35 %	167
4 (MPCI)	100 %	97 %	98 %	0 %	3 %	97 %	1473
5 (MFCI)	97 %	97 %	19 %	3 %	3 %	31 %	124
6 (DoS)	100 %	96 %	99 %	0 %	4%	98 %	391
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	1365

Table 6.23 – Random Forest Gas Pipeline Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	93 %	96 %	94 %	11 %	4 %	95 %	12239
1 (NMRI)	96 %	20 %	32 %	1 %	80 %	24 %	541
2 (CMRI)	98%	93 %	93 %	1 %	7 %	93 %	3103
3 (MSCI)	100 %	93 %	94 %	0 %	11 %	94 %	167
4 (MPCI)	99 %	95 %	97 %	0 %	5 %	96 %	1473
5 (MFCI)	100 %	97 %	97 %	0 %	3 %	97 %	124
6 (DoS)	100 %	96 %	99 %	0 %	4%	97 %	391
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	1365

Table 6.24 – Hybrid DNN ADS Gas Pipeline Multiclass Classification

Class	Accuracy	Recall	Precision	FPR	FNR	F1-score	Support
0 (Normal)	95 %	98 %	94 %	10 %	2 %	96 %	12239
1 (NMRI)	97 %	0 %	0 %	0 %	100 %	0 %	541
2 (CMRI)	99 %	98 %	93 %	1 %	2 %	96 %	3103
3 (MSCI)	100 %	95 %	96 %	0 %	5 %	95 %	167
4 (MPCI)	100 %	98 %	97 %	0 %	2 %	98 %	1473
5 (MFCI)	100 %	94 %	94 %	0 %	6 %	94 %	124
6 (DoS)	100 %	76 %	100 %	0 %	24 %	87 %	391
7 (Recon.)	100 %	100 %	100 %	0 %	0 %	100 %	1365

Table 6.25 – Gas Pipeline Binary Classification

Approach	Accuracy	Recall	Precision	FPR	FNR	F1-score
Decision Tree	78 %	39 %	99 %	0 %	61 %	56 %
Naïve Bayes	81 %	49 %	99 %	0 %	51 %	65 %
Random Forest	93 %	89 %	92 %	4 %	11 %	91 %
Hybrid DNN	95 %	89 %	96 %	2 %	11 %	92 %

6.10 Discussion of Results

6.10.1 Decision Tree Approach

In the Water Storage Tank SCADA dataset, the Decision Tree approach detects all the normal records (100 % of recall) with a 88 % of precision, which leads to a False Positive Rate of 38 %. This is why all the attacks classes 1, 2, 3, 5 and 6 recall is 0 %, because those types of attacks are considered as normal records by the classifier. The binary version of the Water Storage Tank SCADA dataset confirms this analysis with the 62 % detection rate of anomalous records and 38% of False Positive Rate.

The Gas Pipeline SCADA dataset shows the same results for multiclass classification with the Decision Tree approach. Classes 1, 2, 3, 5 and 6 are mis-classified as normal records and we have a 61 % False Positive rate. For binary classification, we have a poor anomaly detection rate of 39 % and a high False Positive Rate of 61 %.

6.10.2 Naïve Bayes Approach

The Naïve Bayes Approach for the Water Storage Tank SCADA dataset has a very low detection rate of normal records (23 %) as well as attack class 1 (0 %). Furthermore, there is a high False Positive Rates for normal records detection (37 %). The 100 % of detection rate for both class 2 and 6 with low precision of 11 % and 3 % respectively is due to normal records being classified as CMRI and DoS attacks. The binary version of the Water Storage Tank dataset gives a detection rate of 100 %, but with a 35 % of False Positive Rate.

On the other hand, The Naïve Bayes Approach for multiclass classification, the Gas Pipeline SCADA dataset has a clear difficulty to classify the normal records (almost 0 % of recall). This is why the attacks classes 1 through 7 have high detection rate with poor precision for classes 1, 3 and 5. Binary classification gives a 100 % of recall for normal records with a high False Positive Rate of 51 %, and a detection rate for anomalous records of 49 %.

6.10.3 Random Forest Approach

When doing a multiclass classification on the Water Storage Tank SCADA dataset, with the Random Forest approach, apart attack classes 1 and 2 where the detection rate is 0, this approach gives better detection rate for the other attacks categories (> 96 %) and the normal records category (100 %). We have the 89 % of precision for normal records detection and a False Positive Rate of 34 %. This is because, attacks classes 1 and 2 classified as normal records. Binary classification of the Water Storage Tank dataset confirms the previous results with a 66 % recall for anomalous records and a False Positive Rate of 34 % for normal records.

For the multiclass classification, the Gas Pipeline SCADA dataset gives a 96 % of detection rate for normal records and a False Positive Rate of 11 %. Apart from the attack class 1 which has a recall of 20 %, all the other classes of attacks (2 through 7) have a recall above 93 % and a False Positive Rate less or equal to 1 %. The binary classification of the Gas Pipeline dataset confirm those results with a 96 % recall for normal records with a False Positive Rate of 11 %, and a 89 % recall for anomalous records.

6.10.4 Comparison of the Proposed Approach with the Baseline Algorithms

Water Storage Tank SCADA dataset

For the Water Storage Tank SCADA dataset, based on the F1-score performance, the Decision Tree and Naïve Bayes have overall bad results compared to Random Forest and the proposed approach in multiclass classification. We will focus our comparison on the proposed approach and Random Forest. The proposed approach and Random Forest are totally mis-classifying class 1 and 2 attacks as those classes have a 0 % of recall and 100 % of False Negative Rate. Reconnaissance attacks (class 7) is fully detected by both Random Forest and the proposed approach (recall of 100 %). Random Forest performs slightly better than the proposed approach on class 3 (96 % and 92 % of recall respectively). Random Forest has higher detection rate than the Hybrid DNN approach on class 4 and

class 5 (98 % vs 87 %, and 100 % vs 85 % of recall respectively). The DoS attack is still poorly detected by our approach compared to Random Forest (35 % and 100 % of recall respectively). The binary classification of the same dataset shows that The proposed approach and Random Forest give the best results (66 % of recall and 34 % of False Negative Rate) compared to the Decision Tree and Naïve Bayes approaches.
In overall, apart from one attack class, the proposed approach has comparable results with the best of the standard algorithms which is the Random Forest.

Gas Pipeline SCADA dataset

For multiclass classification with Gas Pipeline dataset, Decision Tree and Naïve Bayes also show poor results compared to Random Forest and the proposed approach. The proposed approach outperforms Random Forest for class 2, 3 and 4 attacks (98 %, 95 % and 98 % versus 93 %, 93 % and 95 % of recall respectively). The DoS attack recall for the proposed approach, though lower than the one of Random Forest (76 % and 96 % respectively) has increased with this dataset. The class 1 is still not detected by our approach (0 % of recall), but the the detection rate of Random Forest is also still poor for this class (20 % of recall). The binary classification for the Gas Pipeline SCADA dataset, shows that the proposed approach is outperforming all the standard approaches. The Hybrid DNN approach has higher f1-score compared to Random Forest (92 % and 91 % respectively), and has a lowest False Positive Rate compared to Random Forest (2 % and 4 % respectively).

6.10.5 Single vs Distributed Training Time

Figures 6.20 and 6.25 show a clear reduction of the training time when using a cluster of computer for deep models training instead of a single computer. One of the big challenge of deep learning models is the high training time required to train those models. By using a distributed approach, implemented with the TensorFlow distributed framework, the training time has been significantly reduced.

6.11 Conclusion

An openstack infrastructure has been setup to conduct the different experiments of this thesis. A standalone instance is used to implement the hybrid deep neural network anomaly detection system, whereas a Hadoop cluster has been deployed for the distributed approach. The water storage tank and gas pipeline SCADA datasets have been pre-processed i.e. minimized, balanced, one-hot encoding of the labels, and splitted into training, test and validation sets. The proposed approach is compared with the Decision Tree, Naive Bayes and Random Forest standard approaches. In all cases of the tests, the proposed approach either outperforms all the baseline approaches or performs almost as well as the best of them which is the Random Forest. The distributed approach of the deep learning based anomaly detection shows a significant reduction of the training time.

Deep learning approaches are successfully used in domains such as image, video or natural language processing. The research community is trying to apply deep learning approaches to information security, particularly in SCADA systems security. The different results show that deep learning is a promising field to develop efficient anomaly detection systems to protect SCADA networks. The results globally outperform those of baseline approaches. Furthermore, the results also show that the training time challenge of deep learning models could be mitigated with a distributed approach.

Designing deep learning based approach for SCADA systems gives promising results. However, given some of the SCADA networks requirements such as low latency, availability, reliability and performance, trade-offs should be made between those requirements and the gain in terms of detection rate, false positive rate and the training time.

General Conclusion

Review of research

SCADA systems are more and more targeted by cyber-attacks because of vulnerabilities in hardware, software, protocols and communication stack. Nowadays, those systems use standard hardware, software, operating systems and protocols. Furthermore, SCADA systems which used to be air-gaped are now interconnected to corporate networks and to the Internet. To thwart those attacks, many solutions have been proposed such as use of firewalls, anti-viruses, encryption, etc. But due to the differences between traditional IT and SCADA systems, the proposed solutions are not always applicable to the latter. Intrusion detection systems (IDS) are complementary solutions to secure SCADA networks. In the present thesis, we have proposed an anomaly-based intrusion detection system using a deep learning approach. Deep Learning have been successfully used in other research areas like image, video or natural language processing. Our objective through this thesis to propose an accurate anomaly detection system in terms of detection rate and false positive rate, and efficient in terms of processing time in SCADA networks, using an unsupervised deep feature learning approach.

To this end, the proposed approach answered the two following research questions :

RQ1 : *How can we design a deep learning based approach for unsupervised feature learning in SCADA systems ?*

RQ2 : *How can we design a deep learning based framework for efficient and accurate anomaly detection in SCADA systems?*

The first research question is divided into three sub-questions:

- **RQ 1.1 :** *What is the state of the art of deep learning based unsupervised feature learning in SCADA systems ?*
  To answer this question, we have conducted a review of existing deep feature learning based anomaly detection systems for SCADA networks. We found out that various deep neural network approaches were combined to learn most salient features of the SCADA network data, which help to improve detection performances. In many cases, those approaches outperform other anomaly detection approaches. However, the high training time of the deep learning based anomaly detectors remains a big challenge in those approaches.

- **RQ 1.2 :** *What are the characteristics of the SCADA dataset used ?*
  The physical process and the resulting SCADA dataset were studied to answer this question. The physical process is a water tank system controlled by a SCADA network. The dataset obtained from the SCADA network has normal records and seven different types of attacks. The study of the distribution of the data showed that the SCADA dataset is highly imbalanced as anomalous records are fewer compared to normal ones.

- **RQ 1.3 :** *How can we design a deep learning based unsupervised feature learning framework that will learn important features from a SCADA dataset?*

  To answer this question, we built a stacked sparse denoising autoencoder for an unsupervised learning of the features of the SCADA network data. Some noise are added to the SCADA input data, and the building block of the unsupervised deep network are denoising autoencoders. A sparsity constraint is added to the deep network loss function to allow more robust feature learning. The unsupervised deep feature learner is trained one

layer at a time by using an unsupervised greedy layer-wise pre-training scheme.

The second research question has three sub-questions as well:

- **RQ 2.1 :** *How do we build a deep learning general framework for anomaly detection in SCADA systems?*

  As a response to this question, we designed the general framework of the deep learning based anomaly detection system for SCADA networks. The framework has two main parts i.e a pre-processing engine and an anomaly detection engine. The pre-processing engine is responsible for the splitting of the dataset in training, validation and test data sub-sets. It also is responsible of the data normalization, the data balancing and one-hot encoding of the labels. The anomaly detection engine encompasses the unsupervised feature learner previously designed, and a classification layer added to it.

- **RQ 2.2 :** *Using the unsupervised deep feature learning approach of the previous research question, can we build an efficient and accurate classifier for anomaly detection in SCADA networks?*
  The actual design of the detection engine is the answer to that question. A softmax layer is added to the unsupervised deep feature learner to form the overall architecture of the anomaly detection system. The architecture is an hybrid deep neural network anomaly detection system for SCADA networks as there is an unsupervised feature learning followed by a supervised anomaly detection. We have defined the different parameters, hyperparameters, activation and loss functions, and the algorithms used to train the model.

- **RQ 2.3 :** *How can we boost the computational efficiency for deep neural network based anomaly detection system in SCADA networks?*
  One of the main challenge in deep learning is the required high training

time. To answer this question, we proposed a distributed architecture of the approach consisting in a parameter server and worker nodes. The parameter server is responsible for storing the model, distributing it to workers and updating the model, whereas the worker nodes are taking care of the bulk of the computation i.e. the calculation of the gradient during the back-propagation training algorithm. The distributed hybrid deep neural network anomaly detection system has been implemented with the Distributed TensorFlow framework running on an Hadoop cluster.

Review of contributions

Our first contribution is a review of deep neural network-based SCADA anomaly detection systems using feature learning approaches. Then, as a second contribution, we proposed a stacked sparse denoising autoencoder architecture as feature learner for SCADA networks. This framework is able to automatically learn the salient features of the SCADA data that will later be used for classification purpose. A framework for an hybrid deep neural network SCADA anomaly detection system which has a data pre-processing engine and an anomaly detection engine is proposed as the third contribution. To speed-up the training process of the approach, we proposed as a fourth contribution a distributed approach of the deep learning based anomaly detection system, which uses a parameter server and worker nodes. Finally, our fifth contribution is an implementation of the hybrid deep neural network SCADA anomaly detection system with the TensorFlow framework proved that this approach gives better results in terms of detection rate and false alarm rate compared to baseline algorithms such as Decision Tree, Naïve Bayes and Random Forest. For the implementation of the distributed approach, we used the Distributed TensorFlow framework on a Hadoop cluster. The results proved that the training time of the distributed approach is significantly reduced compared to the single machine implementation.

Limitations

The detection rates of the proposed approach for NMRI, CMRI and DoS attack types are lower than other attack types. A possible reason is that those attacks are treated as noise that occurred during the normal operation of the SCADA system. This problem could be mitigated by collecting more training data in order for the system to be able to capture the intrinsic representation of those class of attacks. Features of the SCADA datasets which better describe the measurements of observations could also be developed to better represent the underlined physical process.

Future work

As future work, we would collect larger-scale SCADA datasets from the owner to further test the proposed approach. We expect the approach will perform better with larger datasets, especially with more data related to the NMRI, CMRI and DoS attack types.
Another future direction is also to combine our unsupervised learning framework with architectures such as RNN or LSTM in order to take into account the time series aspects of the SCADA data.
The distributed approach of the anomaly detection system was tested on a cluster which has only classic CPUs. To better evaluate the impact on the training time of the model, we will consider using workers equipped with GPU processors.
Detecting anomalies in real-time in SCADA systems is a primary goal to achieve. The approach should be tested with live data streams, by adding stream data acquisition modules to the framework, as well as tested in a SCADA faulty environment to check if the operational anomalies are detected.
It would also be interesting to submit the proposed anomaly detector to real-world SCADA data in order to test its performances in a real-world environment.
Finally, the water tank and the gas pipeline datasets contain seven categories of attacks. In the future, a thorough modelization of the SCADA systems attack types will help improve the performance of the approach against Advanced Persistent Threats (APT) or unknown attack types.

Bibliography

[1] *7 Types of Neural Network activation functions, How to choose?* https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/. Accessed: 2019-07-30. 2019.

[2] Martın Abadi et al. "TensorFlow: Large-scale machine learning on heterogeneous systems". In: *Software available from tensorflow. org* 1.2 (2015).

[3] *Activation functions and it's types-Which is better?* https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f. Accessed: 2019-07-30. 2017.

[4] Md Zahangir Alom et al. "A state-of-the-art survey on deep learning theory and architectures". In: *Electronics* 8.3 (2019), p. 292.

[5] Erza Aminanto and Kwangjo Kim. "Deep learning in intrusion detection system: An overview". In: *2016 International Research Conference on Engineering and Technology (2016 IRCET)*. Higher Education Forum. 2016.

[6] Muhamad Erza Aminanto et al. "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection". In: *IEEE Transactions on Information Forensics and Security* 13.3 (2018), pp. 621–636.

[7] Leonardo Aniello et al. "Big data in critical infrastructures security monitoring: Challenges and opportunities". In: *arXiv preprint arXiv:1405.0325* (2014).

[8] Stefan Axelsson. "Intrusion Detection Systems: A Survey and Taxonomy". In: *Chalmers University of Technology Technical Report* (Apr. 2000).

[9] David Bailey and Edwin Wright. *Practical SCADA for industry*. Elsevier, 2003.

[10] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. "Towards periodicity based anomaly detection in SCADA networks". In: *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. IEEE. 2012, pp. 1–4.

[11] Boldizsár Bencsáth et al. "The cousins of stuxnet: Duqu, flame, and gauss". In: *Future Internet* 4.4 (2012), pp. 971–1003.

[12] Yoshua Bengio et al. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[13] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. "Unsupervised feature learning and deep learning: A review and new perspectives". In: *CoRR, abs/1206.5538* 1 (2012), p. 2012.

[14] Yoshua Bengio, Yann LeCun, et al. "Scaling learning algorithms towards AI". In: *Large-scale kernel machines* 34.5 (2007), pp. 1–41.

[15] Yoshua Bengio et al. "Greedy layer-wise training of deep networks". In: *Advances in neural information processing systems*. 2007, pp. 153–160.

[16] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[17] Richard J Bolton, David J Hand, et al. "Unsupervised profiling methods for fraud detection". In: *Credit Scoring and Credit Control VII* (2001), pp. 235–255.

[18] William Bolton. *Programmable logic controllers*. Newnes, 2015.

[19] Rodolfo Bonnin. *Building Machine Learning Projects with TensorFlow*. Packt Publishing Ltd, 2016.

[20] Y-lan Boureau, Yann L Cun, et al. "Sparse feature learning for deep belief networks". In: *Advances in neural information processing systems*. 2008, pp. 1185–1192.

[21] Stuart A Boyer. *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[22] Eric Byres. "The air gap: SCADA's enduring security myth". In: *Communications of the ACM* 56.8 (2013), pp. 29–31.

[23] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. "Research Challenges for the Security of Control Systems." In: *HotSec*. 2008.

[24] *Cas pratique, la cybersécurité des systèmes industriels*. https://www.ssi.gouv.fr/uploads/IMG/pdf/Cas_pratique_version_finale-2.pdf/. Accessed: 2019-07-30. 2012.

[25] Craig Chambers et al. "FlumeJava: easy, efficient data-parallel pipelines". In: *ACM Sigplan Notices* 45.6 (2010), pp. 363–375.

[26] Rodrigo Chandia et al. "Security strategies for SCADA networks". In: *International Conference on Critical Infrastructure Protection*. Springer. 2007, pp. 117–131.

[27] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[28] Manuel Cheminod, Luca Durante, and Adriano Valenzano. "Review of security issues in industrial networks". In: *IEEE Transactions on Industrial Informatics* 9.1 (2012), pp. 277–293.

[29] Steven Cherry and R Langner. "How Stuxnet is rewriting the cyberterrorism playbook". In: *Computerworld* (2010).

[30] Steven Cheung et al. "Using model-based intrusion detection for SCADA networks". In: *Proceedings of the SCADA security scientific symposium*. Vol. 46. Citeseer. 2007, pp. 1–12.

[31] Trishul Chilimbi et al. "Project adam: Building an efficient and scalable deep learning training system". In: *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 2014, pp. 571–582.

[32] Clarence Chio and David Freeman. *Machine Learning and Security: Protecting Systems with Data and Algorithms*. " O'Reilly Media, Inc.", 2018.

[33] Gordon R Clarke, Deon Reynders, and Edwin Wright. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004.

[34] *Common Loss functions in machine learning*. https://towardsdatascience.com / common - loss - functions - in - machine - learning - 46af0ffc4d23. Accessed: 2019-07-30. 2018.

[35] A Creery and EJ Byres. "Industrial cybersecurity for power system and SCADA networks". In: *Record of Conference Papers Industry Applications Society 52nd Annual Petroleum and Chemical Industry Conference*. IEEE. 2005, pp. 303–309.

[36] David E Culler. "Dataflow architectures". In: *Annual review of computer science* 1.1 (1986), pp. 225–253.

[37] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 8609–8613.

[38] Tudorica Daniela. "Communication security in SCADA pipeline monitoring systems". In: *2011 RoEduNet International Conference 10th Edition: Networking in Education and Research*. IEEE. 2011, pp. 1–5.

[39] Jeffrey Dean et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012.

[40] Biplob Debnath et al. "BloomFlash: Bloom filter on flash-based storage". In: *2011 31st International Conference on Distributed Computing Systems*. IEEE. 2011, pp. 635–644.

[41] Li Deng, Dong Yu, et al. "Deep learning: methods and applications". In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387.

[42] Dorothy E Denning. "Cyberterrorism: The logic bomb versus the truck bomb". In: *Global Dialogue* 2.4 (2000), p. 29.

[43] Asimenia Dimokranitou. "Adversarial autoencoders for anomalous event detection in images". PhD thesis. Purdue University, Indianapolis, Indiana: Purdue University, 2017.

[44] Abebe Abeshu Diro and Naveen Chilamkurti. "Distributed attack detection scheme using deep learning approach for Internet of Things". In: *Future Generation Computer Systems* 82 (2018), pp. 761–768.

[45] Rob A Dunne and Norm A Campbell. "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function". In: *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*. Vol. 181. Citeseer. 1997, p. 185.

[46] Dacfey Dzung et al. "Security for industrial communication systems". In: *Proceedings of the IEEE* 93.6 (2005), pp. 1152–1177.

[47] Samuel East et al. "A Taxonomy of Attacks on the DNP3 Protocol". In: *International Conference on Critical Infrastructure Protection*. Springer. 2009, pp. 67–81.

[48] Nicolas Falliere, Liam O Murchu, and Eric Chien. "W32. stuxnet dossier". In: *White paper, Symantec Corp., Security Response* 5.6 (2011), p. 29.

[49] Sharifah Yaqoub A Fayi. "What Petya/NotPetya ransomware is and what its remidiations are". In: *Information Technology-New Generations*. Springer, 2018, pp. 93–100.

[50] Cheng Feng, Tingting Li, and Deeph Chana. "Multi-level anomaly detection in industrial control systems via package signatures and lstm networks". In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2017, pp. 261–272.

[51] Ming Feng and Hao Xu. "Deep reinforecement learning based optimal defense for cyber-physical system in presence of unknown cyber-attack". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2017, pp. 1–8.

[52] Ugo Fiore et al. "Network anomaly detection with the restricted Boltzmann machine". In: *Neurocomputing* 122 (2013), pp. 13–23.

[53] Igor Nai Fovino et al. "An experimental platform for assessing SCADA vulnerabilities and countermeasures in power plants". In: *3rd International Conference on Human System Interaction*. IEEE. 2010, pp. 679–686.

[54] Wei Gao. *Cyberthreats, attacks and intrusion detection in supervisory control and data acquisition networks*. PhD Thesis. Mississippi State University, 2013.

[55] Wei Gao et al. "On SCADA control system command and response injection and intrusion detection". In: *eCrime Researchers Summit (eCrime), 2010*. IEEE. 2010, pp. 1–9.

[56] Iñaki Garitano, Roberto Uribeetxeberria, and Urko Zurutuza. "A review of SCADA anomaly detection systems". In: *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*. Springer. 2011, pp. 357–366.

[57] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

[58] Andrew Ginter. *SCADA Security What is broken and how to fix it*. Abterra Technologies Inc., 2016.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[60] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[61] Andy Greenberg. "The Untold story of notpetya, The Most Devastating cyberattack in history". In: *Wired, August* (2018).

[62] Tim Greene. "Experts hack power grid in no time". In: *NetworkWorld* (2008).

[63] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: Concepts and techniques*. Elsevier, 2012.

[64] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[65] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 1322–1328.

[66] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[67] Youbiao He, Gihan J Mendis, and Jin Wei. "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism". In: *IEEE Transactions on Smart Grid* 8.5 (2017), pp. 2505–2516.

[68] Geoffrey E Hinton. "A practical guide to training restricted Boltzmann machines". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 599–619.

[69] Geoffrey E Hinton. "Deep belief networks". In: *Scholarpedia* 4.5 (2009), p. 5947.

[70] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[71] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[72] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: theory and applications". In: *Neurocomputing* 70.1-3 (2006), pp. 489–501.

[73] Peter Huitsing et al. "Attack taxonomies for the Modbus protocols". In: *International Journal of Critical Infrastructure Protection* 1 (2008), pp. 37–44.

[74] *Hyperparameters in Deep Learning*. https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd. Accessed: 2019-09-05. 2019.

[75] *Hyperparameters: Optimization Methods and Real World Model Management*. http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/. Accessed: 2019-07-30. 2019.

[76] *Hyperparameters: Optimization Methods and Real World Model Management*. https://missinglink.ai/guides/neural-network-concepts/hyperparameters-optimization-methods-and-real-world-model-management/. Accessed: 2019-07-30. 2019.

[77] Michael Isard et al. "Dryad: distributed data-parallel programs from sequential building blocks". In: *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*. 2007, pp. 59–72.

[78] Anil Jain, Karthik Nandakumar, and Arun Ross. "Score normalization in multimodal biometric systems". In: *Pattern recognition* 38.12 (2005), pp. 2270–2285.

[79] Katarzyna Janocha and Wojciech Marian Czarnecki. "On loss functions for deep neural networks in classification". In: *arXiv preprint arXiv:1702.05659* (2017).

[80] Ahmad Javaid et al. "A deep learning approach for network intrusion detection system". In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICS. 2016, pp. 21–26.

[81] Yang Ju and Hui-Gang Zhang. "Design and application of IEC 60870-5-104 telecontrol protocol." In: *Relay* 34.17 (2006), pp. 55–58.

[82] Raogo KABORE, Yvon KERMARREC, and LENCA Philippe. "Revue des systèmes de détection d'anomalies dans les réseaux SCADA et attaques internes". In: *Inforsid 2017*. 2017.

[83] Barry L Kalman and Stan C Kwasny. "Why tanh: choosing a sigmoidal function". In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. Vol. 4. IEEE. 1992, pp. 578–581.

[84] Min-Joo Kang and Je-Won Kang. "Intrusion detection system using deep neural network for in-vehicle network security". In: *PloS one* 11.6 (2016), e0155781.

[85] Jihyun Kim et al. "Long short term memory recurrent neural network classifier for intrusion detection". In: *Platform Technology and Service (PlatCon), 2016 International Conference on*. IEEE. 2016, pp. 1–5.

[86] Jin Kim et al. "Method of intrusion detection using deep neural network". In: *Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on*. IEEE. 2017, pp. 313–316.

[87] Kwangjo Kim, Muhamad Erza Aminanto, and Harry Chandra Tanuwidjaja. *Network Intrusion Detection Using Deep Learning: A Feature Learning Approach*. Springer, 2018.

[88] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[89] István Kiss, Béla Genge, and Piroska Haller. "A clustering-based approach to detect cyber attacks in process control systems". In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE. 2015, pp. 142–148.

[90] István Kiss et al. "Data clustering-based anomaly detection in industrial control systems". In: *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 275–281.

[91] Eric D Knapp and Joel Thomas Langill. *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress, 2014.

[92] Moshe Kravchik and Asaf Shabtai. "Detecting cyber attacks in industrial control systems using convolutional neural networks". In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM. 2018, pp. 72–83.

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[94] Ronald L Krutz. *Securing SCADA systems*. John Wiley & Sons, 2005.

[95] Solomon Kullback. *Information theory and statistics*. Wiley, 1959.

[96] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[97] Donghwoon Kwon et al. "A survey of deep learning-based network anomaly detection". In: *Cluster Computing* (2017), pp. 1–13.

[98] Hugo Larochelle and Yoshua Bengio. "Classification using discriminative restricted Boltzmann machines". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 536–543.

[99] Pavel Laskov et al. "Learning intrusion detection: supervised or unsupervised?" In: *International Conference on Image Analysis and Processing*. Springer. 2005, pp. 50–57.

[100] Nicholas Leall. "Lessons from an insider attack on SCADA systems". In: *Online]: http://blogs. cisco. com/security/lessons from an insider attack on scada systems* (2009).

[101] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[102] Robert M Lee, Michael J Assante, and Tim Conway. "Analysis of the cyber attack on the Ukrainian power grid". In: *SANS Industrial Control Systems* (2016).

[103] Liangzhi Li, Kaoru Ota, and Mianxiong Dong. "When weather matters: IoT-based electrical load forecasting for smart grid". In: *IEEE Communications Magazine* 55.10 (2017), pp. 46–51.

[104] Zhipeng Li et al. "Intrusion detection using convolutional neural networks for representation learning". In: *International Conference on Neural Information Processing*. Springer. 2017, pp. 858–866.

[105] Ondrej Linda, Todd Vollmer, and Milos Manic. "Neural network based intrusion detection system for critical infrastructures". In: *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE. 2009, pp. 1827–1834.

[106] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *International Conference on Machine Learning (ICML)*. Vol. 30. 1. 2013, p. 3.

[107] *MAE and RMSE, Which Metric is Better?* https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d. Accessed: 2019-07-30. 2019.

[108] *Maîtriser la SSI pour les systèmes industriels*. https://www.ssi.gouv.fr/uploads/IMG/pdf/Guide_securite_industrielle_Version_finale-2.pdf/. Accessed: 2019-07-30. 2012.

[109] Munir Majdalawieh, Francesco Parisi-Presicce, and Duminda Wijesekera. "DNPSec: Distributed network protocol version 3 (DNP3) security framework". In: *Advances in Computer, Information, and Systems Sciences, and Engineering*. Springer, 2007, pp. 227–234.

[110] Bill Miller and Dale Rowe. "A survey SCADA of and critical infrastructure incidents". In: *Proceedings of the 1st Annual conference on Research in information technology*. ACM. 2012, pp. 51–56.

[111] T.M Mitchell. "Machine Learning, McGraw-Hill Higher Education". In: *McGraw-Hill Science/Engineering/Math* (1997).

[112] *Modbus Application Protocol Specification V1. 1b3*. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. Accessed: 2019-04-12. 2012.

[113] Leila Mohammadpour et al. "A Convolutional Neural Network for Network Intrusion Detection System". In: *Proceedings of the Asia-Pacific Advanced Network* 46 (), pp. 50–55.

[114] David Moore et al. *The spread of the sapphire/slammer worm*. Tech. rep. CAIDA, ICSI, Silicon Defense, UC Berkeley EECS and UC San Diego CSE, 2003.

[115] Thomas Morris and Wei Gao. "Industrial control system traffic data sets for intrusion detection research". In: *International Conference on Critical Infrastructure Protection*. Springer. 2014, pp. 65–78.

[116] Thomas Morris et al. "A control system testbed to validate critical infrastructure protection concepts". In: *International Journal of Critical Infrastructure Protection* 4.2 (2011), pp. 88–103.

[117] Derek G Murray et al. "CIEL: a universal execution engine for distributed data-flow computing". In: *Proc. 8th ACM/USENIX Symposium on Networked Systems Design and Implementation*. 2011, pp. 113–126.

[118] Derek G Murray et al. "Naiad: a timely dataflow system". In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. 2013, pp. 439–455.

[119] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[120] Payam Mahmoudi Nasr and Ali Yazdian Varjani. "Alarm based anomaly detection of insider attacks in SCADA system". In: *Smart Grid Conference (SGC), 2014*. IEEE. 2014, pp. 1–6.

[121] *Neural Networks*. https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html#weights/. Accessed: 2019-07-30. 2019.

[122] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[123] Andrew Nicholson et al. "SCADA security in the light of Cyber-Warfare". In: *Computers & Security* 31.4 (2012), pp. 418–436.

[124] Dong Nie et al. "3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 212–220.

[125] Paul Oman, Edmund Schweitzer, and Deborah Frincke. "Concerns about intrusions into remotely accessible substation controllers and SCADA systems". In: *Proceedings of the Twenty-Seventh Annual Western Protective Relay Conference*. Vol. 160. Citeseer. 2000.

[126] Salima Omar, Asri Ngadi, and Hamid H Jebur. "Machine learning techniques for anomaly detection: an overview". In: *International Journal of Computer Applications* 79.2 (2013).

[127] Santanu Pattanayak, Pattanayak, and Suresh John. *Pro Deep Learning with TensorFlow*. Springer, 2017.

[128] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[129] Evan Perez. "Alleged Russian malware found on Vermont utility's laptop". In: *CNN* (2017).

[130] Siby Jose Plathottam, Hossein Salehfar, and Prakash Ranganathan. "Convolutional Neural Networks (CNNs) for power system big data analysis". In: *2017 North American Power Symposium (NAPS)*. IEEE. 2017, pp. 1–6.

[131] Sasanka Potluri and Christian Diedrich. "Accelerated deep neural networks for enhanced intrusion detection system". In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE. 2016, pp. 1–8.

[132] Sasanka Potluri and Christian Diedrich. "Deep feature extraction for multi-class intrusion detection in industrial control systems". In: *Int. J. Comput. Theory Eng* 9.5 (2017), pp. 374–379.

[133] Z Reitermanova. "Data splitting". In: *WDS*. Vol. 10. 2010, pp. 31–36.

[134] Rosslin John Robles et al. "Vulnerabilities in SCADA and critical infrastructure systems". In: *International Journal of Future Generation Communication and Networking* 1.1 (2008), pp. 99–104.

[135] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[136] Dennis W Ruck et al. "The multilayer perceptron as an approximation to a Bayes optimal discriminant function". In: *IEEE Transactions on Neural Networks* 1.4 (1990), pp. 296–298.

[137] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[138] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[139] Arthur Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–29.

[140] Johannes Schmidt-Hieber. "Nonparametric regression using deep neural networks with ReLU activation function". In: *arXiv preprint arXiv:1708.06633* (2017).

[141] P Sibi, S Allwyn Jones, and P Siddarth. "Analysis of different activation functions using back propagation neural networks". In: *Journal of Theoretical and Applied Information Technology* 47.3 (2013), pp. 1264–1268.

[142] Jill Slay and Michael Miller. "Lessons learned from the maroochy water breach". In: *International Conference on Critical Infrastructure Protection*. Springer. 2007, pp. 73–82.

[143] Jason Stamp et al. "Common vulnerabilities in critical infrastructure control systems". In: *SAND2003-1772C. Sandia National Laboratories* (2003).

[144] Ralf C Staudemeyer. "Applying long short-term memory recurrent neural networks to intrusion detection". In: *South African Computer Journal* 56.1 (2015), pp. 136–154.

[145] Keith Stouffer, Joe Falco, and Karen Scarfone. "Guide to industrial control systems (ICS) security". In: *NIST special publication* 800.82 (2011), pp. 16–16.

[146] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. "Introduction to multi-layer feed-forward neural networks". In: *Chemometrics and intelligent laboratory systems* 39.1 (1997), pp. 43–62.

[147] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[148] Chun Chet Tan and Chikkannan Eswaran. "Performance comparison of three types of autoencoder neural networks". In: *2008 Second Asia International Conference on Modelling & Simulation (AMS)*. IEEE. 2008, pp. 213–218.

[149] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE. 2009, pp. 1–6.

[150] André Teixeira et al. "Attack models and scenarios for networked control systems". In: *Proceedings of the 1st international conference on High Confidence Networked Systems*. ACM. 2012, pp. 55–64.

[151] Chun-Wei Tsai et al. "Big data analytics: a survey". In: *Journal of Big Data* 2.1 (2015), p. 21.

[152] Rose Tsang. "Cyberthreats, vulnerabilities and attacks on SCADA networks". In: *University of California, Berkeley* (2010).

[153] Robert J Turk. *Cyber incidents involving control systems*. Tech. rep. Idaho National Laboratory (INL), 2005.

[154] Alfonso Valdes and Steven Cheung. "Intrusion monitoring in process control systems". In: *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. IEEE. 2009, pp. 1–7.

[155] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103.

[156] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408.

[157] Lidong Wang and Randy Jones. "Big Data Analytics for Network Intrusion Detection: A Survey". In: *International Journal of Networks and Communications* 7.1 (2017), pp. 24–31.

[158] Yong Wang et al. "Srid: State relation based intrusion detection for false data injection attacks in scada". In: *European Symposium on Research in Computer Security*. Springer. 2014, pp. 401–418.

[159] Jin Wei and Gihan J Mendis. "A deep learning-based cyber-physical strategy to mitigate false data injection attack in smart grids". In: *2016 Joint Workshop on Cyber-Physical Security and Resilience in Smart Grids (CPSR-SG)*. IEEE. 2016, pp. 1–6.

[160] Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[161] David Wilson et al. "Deep learning-aided cyber-attack detection in power transmission systems". In: *2018 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE. 2018, pp. 1–5.

[162] Qingyang Xu et al. "The learning effect of different hidden layers stacked autoencoder". In: *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Vol. 2. IEEE. 2016, pp. 148–151.

[163] Weizhong Yan and Lijie Yu. "On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach". In: *Proceedings of the annual conference of the prognostics and health management society*. 2015.

[164] Chuanlong Yin et al. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks". In: *IEEE Access* 5 (2017), pp. 21954–21961.

[165] Mahmood Yousefi-Azar et al. "Autoencoder-based feature learning for cyber security applications". In: *2017 International joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 3854–3861.

[166] Yang Yu, Jun Long, and Zhiping Cai. "Session-based network intrusion detection using a deep learning architecture". In: *International Conference on Modeling Decisions for Artificial Intelligence*. Springer. 2017, pp. 144–155.

[167] Matei Zaharia et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*. 2012, pp. 15–28.

[168] Hongshan Zhao et al. "Anomaly detection and fault analysis of wind turbine components based on deep learning network". In: *Renewable energy* 127 (2018), pp. 825–834.

[169] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. "A taxonomy of cyber attacks on SCADA systems". In: *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*. IEEE. 2011, pp. 380–388.

[170] Bonnie Zhu and Shankar Sastry. "SCADA-specific intrusion detection/prevention systems: a survey and taxonomy". In: *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*. Vol. 11. 2010.

[171] Ciyou Zhu et al. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on Mathematical Software (TOMS)* 23.4 (1997), pp. 550–560.

[172] Richard Zuech, Taghi M Khoshgoftaar, and Randall Wald. "Intrusion detection and big heterogeneous data: a survey". In: *Journal of Big Data* 2.1 (2015), p. 3.

Contents

RÉSUMÉ EN FRANÇAIS

Contexte

Les systèmes SCADA (Supervisory Control And Data Acquisition) sont des systèmes de contrôle industriel utilisés pour l'acquisition et le contrôle de systèmes géographiquement étendues. Ces systèmes sont utilisés dans la distribution d'eau, le traitement des eaux usées, le transport et la distribution d'énergie, les oléoducs et gazoducs, les systèmes de transport en commun, etc. [145]. L'intégration des systèmes SCADA dans la gestion des systèmes industriels permet non seulement d'améliorer les performances, mais également de réduire les coûts d'exploitation [21].

Historiquement, la sécurité n'était pas prise en compte dans la conception des systèmes SCADA. Les concepteurs ont eu recours à deux formes de protection, à savoir le air-gap et la sécurité par l'obscurité. Le premier reposait sur le fait que les réseaux SCADA étaient physiquement isolés des autres réseaux, rendant toute attaque difficile, tandis que le second reposait sur la présomption selon laquelle les informations sur les systèmes SCADA ne sont pas accessibles au public, ce qui les rend ainsi sécurisés [94].

De nos jours, les systèmes SCADA modernes utilisent le matériel et les logiciels commerciaux, ainsi que les technologies de communication standard telles que les protocoles TCP / IP ou sans fil. De plus, les réseaux SCADA actuels sont interconnectés aux réseaux d'entreprise et à Internet pour diverses raisons telles que la gestion, l'administration système, etc. [23] [134].
Ces changements dans les réseaux SCADA, qui permettent une gestion facile et une réduction des coûts, les exposent par contre aux cyber-attaques [125] [21].
Les conséquences d'une attaque sur les systèmes SCADA peuvent être des pertes de production, des pertes financières, des catastrophes environnementales et même des pertes en vies humaines. CiteValdes2009 [169].
Les réseaux SCADA présentent des vulnérabilités dans différents domaines, tels que le périmètre réseau, les protocoles, les appareils de terrain, les bases de données et les liens de communication. Nous pouvons classer les attaques sur les réseaux SCADA de la manière suivante: attaques sur le matériel, attaques sur les logiciels et attaques sur la pile de communications [169] [150].

Après un accès à distance non autorisé aux périphériques SCADA, l'attaquant pourrait injecter de fausses données, telles que la modification des paramètres des automates, ce qui peut causer une défaillance du périphérique ou couper une alarme. L'attaquant peut également modifier la valeur d'affichage de l'IHM

pour tromper l'opérateur.

Les logiciels de réseaux SCADA peuvent être ciblés par des virus, des vers, des chevaux de Troie et des réseaux de zombies. Le smash de pile et la manipulation de pointeurs de fonction pourraient également provoquer un débordement de mémoire tampon et permettre à un attaquant de lancer ses propres programmes contre le système SCADA. En outre, l'accessibilité au web des systèmes SCADA actuels ouvre la porte aux attaques par injection, à l'usurpation du DNS, au détournement de session, au phishing, aux attaques de protocole, aux attaques de la couche application, etc.

La couche réseau peut être attaquée par des attaques de serveur de diagnostic, les scan passifs, la couche de transport par des attaques comme SYN Flood et la couche application par des attaques par modification du DNS ou par injection de commande [169].

De plus, certaines attaques sont spécifiques au protocole SCADA utilisé. Sur le protocole Gould Modicon Modbus [112], les attaquants peuvent effectuer des attaques par réinitialisation de registre, de démarrage à distance ou par reconnaissance d'esclaves. Le détournement de messages broadcast, le rejeu de réponses, le contrôle direct des esclaves, le balayage du réseau, la reconnaissance passive font également partie des attaques possibles sur les protocoles Modbus. Enfin, les systèmes SCADA Modbus TCP pourraient être attaqués en utilisant par utilisation de trames TCP irrégulier, TCP FIN Flood ou épuisement du Pool TCP Pool [73]. D'autres attaques spécifiques à SCADA sont liées au protocole DNP3. Parmi les attaques spécifiques à DNP3, on peut citer la reconnaissance de réseau passive, le rejeu de réponses originales, le dépassement de longueur, réinitialisation de fonction, indisponibilité de fonction, la modification d'adresse de destination, l'interruption de message fragmenté, la modification de la séquence de transport, l'attaque d'écriture de la station distante, la réinitialisation et la configuration des données de la station distante [47]. Ces attaques pourraient avoir un impact sur la disponibilité, la confidentialité et l'intégrité des données des systèmes SCADA et avoir des conséquences telles que le vol d'informations de session, la surveillance et la modification de données SCADA, la divulgation de serveurs, d'ordinateurs et d'informations sur le matériel, etc.

Les réseaux SCADA ont déjà été la cible d'attaques telles que l'attaque sur le système de distribution d'eau de Maroochy en Australie [142], la centrale nucléaire de Davis-Besse dans l'Ohio (États-Unis), Duqu, Flame [110], Stuxnet [48] [29], les centrales électriques en Ukraine [102] et au Vermont aux Etats-Unis

[129].

Les réseaux SCADA sont différents des réseaux informatiques traditionnels et ces spécificités doivent être prises en compte dans les approches de conception de solutions de sécurité pour les systèmes SCADA. Contrairement aux systèmes informatiques traditionnels, les correctifs logiciels et les mises à jour fréquentes ne conviennent pas aux systèmes de contrôle. Ces systèmes requièrent également une haute disponibilité, intègrent un grand nombre de matériel démodés, ont une topologie statique et des modes de communication réguliers [23] [170] [30]. De plus, les systèmes embarqués utilisés pour mettre en œuvre des systèmes d'automatisation industriels souffrent également de restrictions telles que la mémoire, les limitations de puissance de traitement, le manque de robustesse, les problèmes d'implémentation logicielle et de configuration [46].

Pour protéger les réseaux SCADA, [26] propose de nombreuses normes de sécurité propres à SCADA, telles que le déploiement de pare-feu, la surveillance des messages, les solutions basées sur des protocoles, la gestion des clés cryptographiques, les antivirus et les correctifs logiciels. Une politique de sécurité cohérente, une architecture réseau bien conçue, le durcissement du système en fermant les services, les ports et les logiciels inutiles, le mot de passe à deux facteurs et le cryptage des données pour les connexions distantes sont également utiles pour sécuriser les systèmes SCADA [35] [125] propose également l'authentification des partenaires de communication. Les autres solutions utilisées pour protéger les réseaux SCADA sont les protocoles sécurisés et les réseaux privés virtuels (VPN).

Malgré la combinaison de toutes ces mesures de sécurité, il n'ya pas de risque zéro en matière de sécurité des systèmes d'information, c'est-à-dire qu'une attaque pourrait réussir sur le réseau. Un système de détection d'intrusion (IDS) est une solution de sécurité qui joue un rôle important dans la sécurité des informations car elle permet de détecter les intrusions réussies dans un réseau. [56] plaide pour que les systèmes IDS soient utilisés avec d'autres mécanismes de sécurité tels que des pare-feu, des scanners de vulnérabilité, des vérificateurs de politique de sécurité pour assurer une sécurité optimale du réseau de contrôle industriel. Mais la nature spécifique des systèmes SCADA nécessite des approches spécifiques pour les systèmes de détection d'intrusion SCADA.

Il existe principalement trois types d'approches de détection d'intrusion: la détection de signature, la détection d'anomalie et la détection basée sur un modèle. La détection de signature compare le trafic pour connaître la signature.

D'autre part, la détection d'anomalie apprend le comportement "normal" du système et tente de détecter des anomalies dans le trafic, c'est-à-dire un trafic s'écartant du comportement "normal". Les principaux inconvénients de la détection des anomalies sont le faible taux de détection et le nombre élevé de fausses alarmes. Mais les IDS basés sur la détection d'anomalies peuvent détecter des nouvelles attaques. L'approche basée sur un modèle utilise des règles pour créer un modèle de ce qui est autorisé et déclenche des alertes lorsque le comportement observé ne correspond pas aux règles. Bien que cela semble prometteur, il est difficile de modéliser entièrement un système [8] [169]. Par conséquent, la détection des anomalies est un mécanisme de défense important pour protéger les systèmes SCADA.

Objectif et questions de recherche

Au cours des dernières années, le Deep Learning [59], un sous-domaine du Machine Learning, est devenu un sujet brûlant parmi les chercheurs car cette approche est appliquée avec succès à des domaines tels que la classification d'images, la vidéo et le traitement du langage naturel. Dans la littérature, on tente de plus en plus d'utiliser le deep learning pour la détection des anomalies de réseaux [6] [44] [50] [51] [52] [67] [80] [84] [85] [86].

Cependant, très peu d'approches de réseaux deep learning non supervisées ont été utilisées pour la détection d'anomalies dans le domaine spécifique des réseaux SCADA.

Les systèmes de détection d'intrusion basés sur des anomalies sont soit supervisés ou non supervisés. Dans les méthodes supervisées, les données d'apprentissage sont étiquetées "normales" ou "anormales" par l'expert du domaine et le système est entraîné à faire la distinction entre les observations "normales" et "anormales", de sorte que les nouvelles observations puissent être classées comme "normales" ou "anormales". Dans les approches non supervisées, il n'y a pas de données étiquetées. Les comportements "normaux" sont modélisés, de sorte que le système puisse détecter les observations présentant une différence significative par rapport aux comportements "normaux" cite laskov2005learning cite Wang2017. noindent Les techniques d'apprentissage deep learning peuvent utiliser des stratégies non supervisées pour apprendre automatiquement les représentations hiérarchiques dans les architectures deep learning aux fins de classification [20] [59] [92] [97] [104] [105] [113] [130] [132].

Mon objectif de recherche dans cette thèse est de proposer un système de dé-

tection d'anomalies précis en termes de taux de détection et de taux de faux positifs, et efficace en termes de temps de traitement dans les systèmes SCADA, en utilisant une approche deep learning d'apprentissage en profondeur des caractéristiques non supervisé. Pour atteindre notre objectif, nous devons aborder les questions de recherche suivantes:

QR 1 : *Comment pouvons-nous concevoir une approche deep learning pour l'apprentissage non supervisé des caractéristiques dans les systèmes SCADA?*

QR 2 : *Comment pouvons-nous concevoir un système pour la détection efficace et précise des anomalies dans les systèmes SCADA ?*

La première question de recherche est divisée en trois sous-questions :

QR 1.1 : *Quel est l'état de l'art de l'apprentissage deep learning non supervisé des caractéristiques dans les systèmes SCADA?*

QR 1.2 : *Quelles sont les caractéristiques du jeu de données SCADA utilisé?*

QR 1.3 : *Comment pouvons-nous concevoir un système deep learning d'apprentissage des caract"ristiques non supervisé basé qui apprendra les caractéristiques importantes des données SCADA?*

La deuxième question de recherche comporte également trois sous-questions:

QR 2.1 : *Quels sont les éléments constitutifs d'un système de détection d'anomalies deep learning dans les systèmes SCADA?*

QR 2.2 : *Avec l'apprentissage des caractéristiques non supervisé, pouvons-nous créer un classificateur efficace et précis pour la détection des anomalies dans les réseaux SCADA?*

QR 2.3 : *Comment pouvons-nous améliorer l'efficacité du temps de calcul pour le système de détection d'anomalie deep learning dans les réseaux SCADA?*

Revues des contributions

Au cours des travaux de cette Thèse, nous avons réalisé cinq contributions :

Contribution 1
Notre première contribution est une revue des systèmes de détection d'anomalies SCADA deep learning utilisant des approches d'apprentissage de de caractéristiques.

Contribution 2
Pour la deuxième contribution, nous avons proposé une architecture stacked sparse denosing auto-encoder comme méthode d'apprentissage des caractéristiques des données SCADA. Ce système est capable de faire un apprentissage automatique des principales caractéristiques des données SCADA qui seront utilisées ultérieurement à des fins de classification.

Contribution 3
Un système deep learning de détection d'anomalies SCADA hybride, doté d'un moteur de prétraitement des données et d'un moteur de détection d'anomalies, est proposé comme troisième contribution. Dans le moteur de pré-traitement, les données sont normalisées, équilibrées et hot-encodé. Le moteur de détection d'anomalies comporte un module d'apprentissage des caractéristiques non supervisé auquel est ajouté un classificateur supervisé.

Contribution 4
Pour accélérer le processus de formation de notre approche, nous avons proposé comme quatrième contribution une version distribuée qui utilise un serveur de paramètres et des nœuds esclaves. Le serveur de paramètres stocke les paramètres du modèle et les distribue aux nœuds esclaves où toute la charge de travail se produit. Chaque nœud esclave est responsable du calcul du gradient lors de l'algorithme de rétro-propagation. Le gradient calculé par chaque nœud esclave est renvoyé au serveur de paramètres qui met à jour les poids et le transmet à nouveau aux nœuds esclave.

Contribution 5
Enfin, notre cinquième contribution est une implémentation du système hybride de détection d'anomalies deep learning pour les réseaux SCADA avec le framework TensorFlow. Les expérimentations ont prouvé que cette approche faisait jeu égal avec les meilleurs parmi les approches standards quand elle ne les surclassait pas dans d'autres cas en termes de taux de détection et de taux de faux positifs. La version distribuée de notre modèle permet de réduire considérablement le temps d'entraînement des modèles deep learning.

Hybrid Deep Neural Network Anomaly Detection System for SCADA Networks

Abstract

SCADA systems are more and more targeted by cyber-attacks because of vulnerabilities in hardware, software, protocols, communication stack. Those systems nowadays use standard hardware, software, operating systems and protocols. Furthermore, SCADA systems which used to be air-gaped are now interconnected to corporate networks and to the Internet. To thwart those attacks, many solutions have been proposed such as use of firewalls, anti-viruses, encryption, etc. But due to the differences between traditional IT and SCADA systems, the proposed solutions are not always applicable to the latter. Intrusion detection systems (IDS) are complementary solutions to secure SCADA networks. Collecting and labeling huge SCADA data is not always feasible as it requires human expert intervention and it is a time consuming process. In recent years Deep learning research has become a hot topic, and sound results have been proposed in image, video and Natural Image Processing. Deep Learning models have the capability to automatically learn features from data in an unsupervised manner. But a big challenge in deep learning model is the high training time of its models. Our thesis objective is to propose an accurate anomaly detection system in terms of detection rate and false positive rate, and efficient in terms of processing time in SCADA networks, using an unsupervised deep feature learning approach. We are using the automatic feature learning capability to unsupervisingly learn SCADA data in order to classify them into normal or anomalous data. We build an hybrid deep neural network anomaly detection system for a Water Tank and Gas Pipeline SCADA systems. Our anomaly detection system is composed of a stacked denoising autoencoder unsupervised feature learner and a softmax classifier. Afterward, we proposed a distributed version of our approach in an attempt to lessen the training time of our deep learning models. The distributed approach is implemented with the TensorFlow Distributed Framework and uses a parameter server to store the model parameters, update them and transmit them to worker nodes which are responsible for calculating the parameters gradient requiring high matrix multiplication during the back-propagation algorithm. Each worker node transmits the calculated gradient to the parameter server for update. Our Hybrid deep neural network anomaly detection system have been compared to standards algorithms. The experimentations proved that the proposed approach compete equally with the best among the baseline algorithm or outperform them in other cases. Furthermore, the distributed version of the proposed approach significantly lowers the training time of our deep learning models.

Keywords: cybersecurity, deep leaning, intrusion detection system, scada

---

LUSSI / Lab-STICC UMR CNRS 6285 – UBL
Technopole Brest Iroise – F-29238 Brest cedex – France

Titre : Réseaux Neuronaux Profonds Hybrides de Détection d'Anomalies pour les Systèmes SCADA

Mots clés : Cyber sécurité, SCADA, Systèmes de Contrôle Industriels, Deep Learning, Machine Learning, Système de Détection d'Anomalies

Résumé : Les systèmes SCADA sont de plus en plus ciblés par les cyberattaques en raison de nombreuses vulnérabilités dans le matériel, les logiciels, les protocoles et la pile de communication. Ces systèmes utilisent aujourd'hui du matériel, des logiciels, des systèmes d'exploitation et des protocoles standard. De plus, les systèmes SCADA qui étaient auparavant isolés sont désormais interconnectés aux réseaux d'entreprise et à Internet, élargissant ainsi la surface d'attaque.
Dans cette thèse, nous utilisons une approche deep learning pour proposer un réseau de neurones profonds hybride efficace pour la détection d'anomalies dans les systèmes SCADA.
Les principales caractéristiques des données SCADA sont apprises de manière automatique et non supervisée, puis transmises à un classificateur supervisé afin de déterminer si ces données sont normales ou anormales, c'est-à-dire s'il y a une cyber-attaque ou non. Par la suite, en réponse au défi dû au temps d'entraînement élevé des modèles deep learning, nous avons proposé une approche distribuée de notre système de détection d'anomalies afin de réduire le temps d'entraînement de notre modèle.

Title : Hybrid Deep Neural Network Anomaly Detection System for SCADA Networks

Keywords : Cybersecurity, SCADA, Industrial Control Systems, Deep Leaning, Machine Learning , Anomaly Detection System

Abstract: SCADA systems are more and more targeted by cyber-attacks because of many vulnerabilities in hardware, software, protocols and the communication stack. Those systems nowadays use standard hardware, software, operating systems and protocols. Furthermore, SCADA systems which used to be air-gaped are now interconnected to corporate networks and to the Internet, widening the attack surface.
In this thesis, we are using a deep learning approach to propose an efficient hybrid deep neural network for anomaly detection in SCADA systems.
The salient features of SCADA data are automatically and unsupervisingly learnt, and then fed to a supervised classifier in order to dertermine if those data are normal or abnormal, i.e if there is a cyber-attack or not. Afterwards, as a response to the challenge caused by high training time of deep learning models, we proposed a distributed approach of our anomaly detection system in order lo lessen the training time of our model..
.