



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

EIT

FAKULTÄT FÜR
ELEKTROTECHNIK UND
INFORMATIONSTECHNIK

Efficient Deep Learning Algorithms for Securing Industrial Control Systems from Cyberattacks

Dissertation

zur Erlangung des akademischen Grades

**Doktoringenieur
(Dr.-Ing.)**

von: Dipl.-Ing. Sasanka Potluri

geboren am: 15.04.1988 in: Vijayawada, Indien

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr.-Ing. Christian Diedrich
Prof. Dr. rer. nat. Andreas Wendemuth

Promotionskolloquium am 26.03.2021

Intentionally left blank

"AI has always been – to my mind – a subject whose potential is on par with some of other fundamental historic scientific achievements like say quantum physics or logic, possibly however with much more dramatic applications and influence on how we will live on this planet in the future. Recent breakthroughs in neural computing and deep learning triggered a new excitement worldwide, but it is now becoming increasingly clear, that these techniques alone will not achieve machine intelligence on par with human capabilities. Reasoning on a symbolic level, as we humans do and which has been the hallmark of classic (good old fashion) AI research, has to be part of it. The scientific challenge of the immediate future is how to integrate these two research paradigms" [1].

— Prof Dr. (Ph D) grad Ing Jörg Siekmann, EurAI and GI Fellow, Co-Founder of DFKI, Senior Professor at University of the Saarland and DFKI, Germany

"Recent breakthroughs in deep learning have created the illusion that deep learning will solve all of AI. This illusion is false as truly intelligent machines should not only learn but also reason, just like humans do. Self-driving cars, for instance, must comply with, and hence, reason about traffic regulations. The long-term promise of AI can only be realised by fostering all AI techniques. A too narrow focus is like claiming that the important invention of rubber tires in the automobile industry could also lead to better engines" [1].

— Luc De Raedt, EurAI Fellow, ERC Advanced Grant, KU Leuven, Belgium.

Intentionally left blank

Dedicated to my wife Jyothsna and my parents Krishna Prasad and Padmaja.

Abstract

Modern Industrial Control System (ICS) represent a wide variety of networked infrastructure connected to the physical world. Depending on the application, these control systems are termed as Process Control Systems (PCS), Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS) or Cyber Physical Systems (CPS). Nowadays, the internet has been evolved as a universal communication platform in many domains, including ICS. The major technical background of the latest industrial revolution (Industrie 4.0 or Smart Factories) is the introduction of internet technologies into the industry making the field devices, machines, plants and factories connected to a network. As ICS is designed for reliability; but security especially against cyber threats is also a critical need. Despite several measures, every day a new attack against the ICS is being identified. Therefore, a proper measure is necessary to identify those novel attacks and ensure security.

Cybersecurity through detection of malicious activities in ICS by efficiently configuring the deep learning algorithms is the main research foci of this thesis. Through research, the cyber-attacks on ICS can be broadly classified as network attacks or injection attacks. In order to develop the deep learning-based cybersecurity, a proper dataset providing the possible attacks on an ICS is necessary. For network attacks, different datasets do exist. Out of them, NSL-KDD is popularly used by many researchers and is selected for the development of Intrusion Detection System (IDS) in ICS for network attacks. As no proper dataset exists for injection attacks, a dataset for injection attacks is simulated using the data from process control plant in the institute.

In order to identify the novel or unknown attack, anomaly-based intrusion detection technique is developed using different deep learning algorithms for classification of normal to anomalous behaviour and a proof-of concept was implemented. The implementations are done in MATLAB using different deep learning libraries originally from MATLAB and also from other sources such as Theano, Tensorflow. Despite classifying the malicious behaviour, this thesis also concentrates on the classification of multiple attack classes. The use of deep learning algorithms for cyber security improves the detection accuracies and are efficient in the identification of novel attacks when compared to the existing approaches. Hybrid deep

learning approaches are also proposed and found to be good in identifying the attacks more accurately and improve the detection accuracy during identification of multiple attack classes.

The contribution of this thesis is as follows: identification and configuration of different deep learning algorithms for drawing hidden complex relations between the input dataset and multi-class attack classification were performed and assessed using famous NSL-KDD dataset for network attacks. Deep learning algorithms are also used to identify complex relations between traditional features and use them to identify injection attacks possible on ICS and their detection accuracies was assessed. Finally, with the outcome of thesis results, development of special injection attack toolbox is developed so that in future researchers can use this toolbox in development of more complex defence in depth strategies for injection attacks against ICS.

Keywords: Deep Learning, Network attacks NSL-KDD, Injection attacks, Industrial Control Systems, Multi-class classification, Intrusion Detection System, anomaly detection, Cyber Security, Network security

Intentionally left blank

Moderne Industrial Control System (ICS) repräsentieren eine Vielfalt von vernetzten Infrastrukturen, die mit der physikalischen Welt verbunden sind. Je nach Anwendung werden diese Control Systeme als Process Control Systems (PCS), Supervisory Control and Data Acquisition (SCADA) Systeme, Distributed Control Systems (DCS) oder Cyber Physical Systems (CPS) bezeichnet. Der wichtigste technische Hintergrund der neuesten industriellen Revolution (Industrie 4.0 oder Smart Factories) ist die Einführung von Internet-Technologien in der Industrie, die die Feldgeräte, Maschinen, Anlagen und Fabriken mit einem Netzwerk verbinden können. Da ICS auf Zuverlässigkeit ausgelegt sind, ist aber auch die Sicherheit, insbesondere gegen Cyberangriffe, ein kritisches Erfordernis. Trotz mehrerer Maßnahmen, wird täglich ein neuer Angriff auf das ICS identifiziert. Daher ist eine angemessene Maßnahme ist notwendig, um diese neuartigen Angriffe zu identifizieren und die Sicherheit zu gewährleisten.

Cybersicherheit durch Erkennung bösartiger Aktivitäten im ICS durch effiziente Konfiguration der Deep-Learning-Algorithmen ist der Schwerpunkt in dieser Arbeit. Durch Forschung können die Cyber-Angriffe auf das ICS allgemein als Netzwerkangriffe oder Injektionsangriffe klassifiziert werden. Um die auf Deep Learning basierende Cybersicherheitsstrategie zu entwickeln, ist ein geeigneter Datensatz notwendig, der die möglichen Angriffe auf ein ICS bereitstellt. Für Netzwerkangriffe gibt es verschiedene Datensätze. Aus diesen wird NSL-KDD von vielen Wissenschaftlern gerne verwendet und für die Entwicklung des Intrusion Detection System (IDS) im ICS für Netzwerkangriffe ausgewählt. Da es keinen eigenen Datensatz für Injektionsangriffe gibt, wird ein Datensatz für Injektionsangriffe mit den Daten aus der Prozesskontrollanlage im Institut simuliert.

Um den neuen oder unbekanntem Angriff zu identifizieren, wird eine Anomalie basierte Intrusion Detection-Technik entwickelt, die durch verschiedene Deep-Learning-Algorithmen zur Klassifizierung von normalem und anomalem Verhalten verwendet und ein Proof-of-Konzept implementiert. Die Implementierungen in MATLAB erfolgten mit verschiedenen Deep-Learning-Bibliotheken, die ursprünglich aus MATLAB und auch aus anderer Herkunft wie Theano, Tensorflow sind. Trotz der Klassifizierung des bösartigen Verhaltens, diese Arbeit konzentriert sich auch auf die Klassifizierung mehrerer Angriffsklassen. Der Einsatz von Deep-

Learning-Algorithmen für die Cybersicherheit verbessert die Erkennungsgenauigkeit und ist im Vergleich zu den bestehenden Ansätzen effizient bei der Identifizierung neuer Angriffe. Hybride Deep-Learning-Ansätze werden ebenfalls vorgeschlagen und als gut befunden, um die Angriffe genauer zu identifizieren und die Erkennungsgenauigkeit bei der Identifizierung mehrerer Angriffsklassen zu verbessern.

Der Beitrag dieser Arbeit sind wie folgt: Identifizierung und Konfiguration von verschiedenen Deep-Learning-Algorithmen für die Zeichnung versteckter komplexer Beziehungen zwischen dem Input-Datensatz und Multi-Class-Angriff Klassifizierung wurde durchgeführt und bewertet mit Hilfe der berühmten NSL-KDD Datensatz für Netzwerk-Angriffe. Deep-Learning-Algorithmen werden auch verwendet, um komplexe Zusammenhänge zwischen traditionellen Merkmalen zu identifizieren und sie zu nutzen, um mögliche Injektionsangriffe auf das ICS zu identifizieren, und ihre Erkennungsgenauigkeiten wurden bewertet. Abschließend, mit dem Ergebnis der Arbeits, wird eine Entwicklung einer speziellen Injektionsangriffs-Toolbox getan wird, so dass in Zukunft Wissenschaftler diese Toolbox bei der Entwicklung komplexerer Defence-in-Depth Strategien für Injektionsangriffe gegen ICS verwenden können.

Stichworte: Deep Learning, Netzwerkangriffe, NSL-KDD, Injektionsangriffe, Industrial Control System, Multiklassen-Klassifizierung, Intrusion Detection System, Anomalie Erkennung, Cybersicherheit, Netzwerksicherheit

Intentionally left blank

Table of Contents

Abstract	vi
Kurzfassung.....	ix
List of Figures	xvi
List of Tables.....	xix
List of Acronyms.....	xxi
List of Publications.....	xxiv
1. Introduction	27
1.1. Background and Motivation	27
1.2. Problem statement	31
1.3. Research aims	32
1.4. Research contributions	34
1.5. Outline of the document	35
2. Classification of cyber-attacks on ICS	37
2.1. Network attacks	37
2.2. Injection attacks	38
3. Literature review	43
3.1. Background.....	43
3.2. Intrusion detection system.....	44
3.2.1. Classification of intrusion detection system.....	46
3.3. Intrusion Detection Systems – Network Attacks.....	54
3.3.1. Machine learning for NIDS	55
3.3.2. Deep Learning for NIDS	60
3.4. Intrusion detection System - FDIA.....	62
3.5. Evaluation metrics of IDS	65
3.6. Conclusions and limitations of IDS.....	66
4. Dataset.....	68

4.1	Network attack dataset – NSL-KDD	68
4.2.	Network attack dataset – UNSW-NB-15.....	81
4.3.	Injection attack dataset - process control plant.....	93
5.	Concept and usage of deep learning algorithms.....	98
5.1.	Steps of data flow from ICS to attack classification	98
5.2.	Data source	99
5.3.	Pre-processing	99
5.3.1.	Pre-processing of NSL-KDD dataset	100
5.3.2.	Pre-processing of UNSW-ND-15 dataset	101
5.3.3.	Pre-processing injection attack dataset	102
5.3.4.	Pre-processing for implementing CNN.....	102
5.4.	Feature extraction	109
5.4.1.	Feature extraction NSL-KDD dataset	111
5.4.2.	Feature extraction UNSW-ND-15 dataset UNSW-NB-15 dataset	111
5.4.3.	Feature extraction injection attack dataset	111
5.5.	Concepts of Deep Learning algorithms	118
5.5.1.	Stacked Autoencoders	118
5.5.2.	Deep Belief Networks	120
5.5.3.	Convolutional Neural Networks.....	122
5.5.4.	Algorithm configuration parameters	122
5.6.	Usage of Deep Learning algorithms.....	128
5.6.1.	NSL-KDD dataset - Stacked Autoencoder.....	128
5.6.2.	NSL-KDD dataset - Deep Belief Network.....	130
5.6.3.	NSL-KDD dataset - Convolutional Neural Networks.....	131
5.6.4.	UNSW-NB-15 - Stacked Autoencoder	133
5.6.5.	UNSW-NB-15 - Deep Belief Networks.....	135
5.6.6.	UNSW-NB-15 - Convolutional Neural Networks	136
5.6.7.	Injection attack - Stacked Autoencoder	137
5.6.8.	Injection attack - Deep Belief Networks	138
5.6.9.	Injection attack - Convolutional Neural Networks	140
5.7.	Hybrid deep learning models.....	141
5.8.	High performance models	142
5.9.	Attack Classification.....	143
5.9.1.	NSL-KDD dataset	143

5.9.2. UNSW-NB-15 dataset.....	144
5.9.3. Injection attack dataset.....	146
6. Results and evaluation.....	147
6.1. Selected results	147
6.1.1. Deep learning models for network attacks	147
6.1.2. Deep learning models for injection attacks	153
6.1.3. Hybrid deep learning algorithms.....	155
6.1.4. High performance model.....	156
6.2. Evaluation.....	158
7. False data injection attacks toolbox in MATLAB/Simulink.....	162
7.1. Naïve Malicious Response Injection (NMRI) attacks.....	162
7.2. Complex Malicious Response Injection (CMRI) attacks	166
7.3. Malicious State Command Injection (MSCI) attacks.....	169
7.4. Malicious Parameter Command Injection (MPCI) attacks.....	171
7.5. Malicious Function Code Injection (MFCI) attacks.....	172
8. Conclusion and Future work	175
8.1. Conclusion.....	175
8.2. Future work.....	177
References	179
Appendix A	193
Notable Cyber Attacks on Industrial Control Systems.....	193
Appendix B	195
Complete Results.....	195
B.1. NSL-KDD dataset.....	195
B.1.1 Stacked autoencoders	195
B.1.2 Deep belief networks	198
B.2. UNSW-NB-15 dataset	200
B.2.1 Stacked autoencoders	200
B.2.2 Deep belief networks	211

B.3. Injection attack dataset.....	220
B.3.1 Stacked autoencoders	221
B.3.2 Deep belief networks	222
B.4. Convolutional neural networks	223
B.4.1 NSL-KDD dataset.....	224
B.4.2 UNSW-NB-15 dataset	226
B.4.3. Injection attack dataset	237
B.5. Hybrid deep learning.....	238
Appendix C	248
Shallow Neural Networks for Injection Attacks in automation applications	248
Appendix D	259
Simulation of network attacks dataset for future applications	259

List of Figures

Figure 1: Typical ICS architecture 27

Figure 2: CIA triad and causes for their violations 28

Figure 3 : Classification of possible cyber-attacks on ICS 37

Figure 4: Possible placement of an IDS in a network infrastructure 45

Figure 5: Classification of intrusion detection systems 47

Figure 6: Functional placement of NIDS in a network infrastructure..... 48

Figure 7: Functional placement of HIDS in a network infrastructure..... 49

Figure 8: Functional diagram of signature based intrusion detection system 50

Figure 9: Illustration of anomalies in a 2-dimensioanl dataset 51

Figure 10: Architecture of network packet and parameters in network packet used for identification of network attacks 55

Figure 11: Traditioanl architecture of artificial neural networks 56

Figure 12: Archtiecture of decision trees 57

Figure 13: Sample SVM classification in a 2D dataspace 58

Figure 14: Archtiecture of network packet and parameters in network packet used for identification of injection attacks 62

Figure 15: Confusion Matrix to understand TP, TN, FP, FN..... 65

Figure 16: Visualization of no. of records in NSL-KDD datasets 80

Figure 17: Used network protocols by different attack types 81

Figure 18 : The Sample Records of UNSW-NB 15 Train and Test Data 92

Figure 19: Process control plant and P&ID diagram of the plant 94

Figure 20: Tank level sensor data (a) Filtered signal (b) Attack injected signal 95

Figure 21: Pump data (a) Filtered signal (b) Attack injected signal 96

Figure 22: Schema for implementation of deep learning in securing ICS 98

Figure 23: Hot encoder for binary extraction..... 103

Figure 24: Discretizer for Binary Extraction of feature 104

Figure 25: Image representation of NSL-KDD dataset..... 104

Figure 26: Input image 8x8 of different classes from NSL – KDD Dataset..... 105

Figure 27: Image representation of UNSW-ND-15 dataset 106

Figure 28 : Input image 8x8 of different classes from UNS-NB-15 dataset – Set 1..... 107

Figure 29: Input image 8x8 of different classes from UNS-NB-15 dataset – Set 2..... 107

Figure 30: Image representation of injection attack dataset..... 108

Figure 31: Input image 8x8 of level sensor data	109
Figure 32: Input image 8x8 of pump data	109
Figure 33: Use of windowing technique for feature extraction from sensor data.....	112
Figure 34: Extracted feature set 1 on level sensor data (a) mean (b) standard deviation (c) variance (d) median	114
Figure 35: Extracted feature set 2 on level sensor data (a) RMS (b) mean (first differential) (c) mean (second differential) (d) maximum	115
Figure 36: Extracted feature set 3 on level sensor data (a) minimum (b) peak to RMS (c) Kurtosis	115
Figure 37: Extracted feature set 1 on pump data (a) mean (b) standard deviation (c) variance (d) median	116
Figure 38: Extracted feature set 2 on pump data (a) RMS (b) mean (first differential) (c) mean (second differential) (d) maximum	117
Figure 39: Extracted feature set 3 on pump data (a) minimum (b) peak to RMS (c) Kurtosis	117
Figure 40 : Structure of stacker autoencoder	119
Figure 41: Structure of deep belief network.....	121
Figure 42. Structure of convolutional neural network	122
Figure 43: Implementation of SAE on NSL_KDD dataset.....	129
Figure 44: Implementation of DBN on NSL_KDD dataset.....	130
Figure 45: Implementation of CNN on NSL-KDD dataset.....	131
Figure 46: Training and validation curve of CNN for a given dataset.....	133
Figure 47: Implementation of SAE on UNSW-NB-15 dataset	134
Figure 48: Implementation of DBN on UNSW-NB-15 dataset	135
Figure 49: Implementation of CNN on UNSW-NB-15 dataset	136
Figure 50: Implementation of SAE on injection attack dataset	138
Figure 51: Implementation of DBN on injection attack dataset.....	139
Figure 52: Implementation of CNN on injection attack dataset.....	140
Figure 53: Overall detection accuracies of deep learning on NSL-KDD dataset	148
Figure 54: Overall detection accuracies of deep learning on UNSW-NB 15 dataset	151
Figure 55: Benchamrking detection accuracies of NSL-KDD dataset with exisitng daapproaches	152
Figure 56: Overall detection accuracies of deep learning algorithms on measurement injection	153
Figure 57: Overall detection accuracies of deep learning algorithms on command injection	155

Figure 58: Sample recommendation of deep learning security concept on injection attacks	161
Figure 59: MATLAB/Simulink window of attack injection toolbox.....	162
Figure 60: Naïve read payload size attack behavior	163
Figure 61: Invalid read payload size attack behavior.....	164
Figure 62: Naïve false error response attack behavior	165
Figure 63: Sporadic sensor measurement attack behavior	166
Figure 64: Slope sensor measurement attack behavior	167
Figure 65: High slope measurement attack behavior	167
Figure 66: High frequency measurement injection attack behavior	168
Figure 67: Altered system control scheme behaviour.....	170
Figure 68: Altered actuator state behaviour	171
Figure 69: Altered control set points behaviour	172
Figure 70: Force listen only mode behaviour with GUI	173
Figure 71: Restart communication attack behavior.....	174
Figure 72: Manufacturing plant use-case in V-REP	248
Figure 73. GUI based attack injection tool	250
Figure 74. (a) 6-Axis ABB Robot [194] (b) Control command sequence	251
Figure 75. Attack on (a) Position Measurement (b) Sensor Measurement (c) Command sequence	252
Figure 76: Workflow of ANN based FDIA identifier.....	253
Figure 77: ANN based FDIA identification (a) Training phase (b) Testing phase.....	256
Figure 78. Confusion matrix of the Neural Network at best validation performance.....	257

List of Tables

Table 1-1: Priorities of security attributes in IT vs ICS	29
Table 2-1: List of FDIA attacks and their sub-classes [20].....	39
Table 3-1: Comparison of key differences between signature based and anomaly based intrusion detection	53
Table 4-1: Attack types group into their respective attack classes [145]	69
Table 4-2: Basic features present in each network connection vector [145].....	70
Table 4-3 : Content related features present in each network connection vectore [145]	71
Table 4-4: Time related traffic features present in each network connection vector [145]	73
Table 4-5: Host based traffic features present in a network connection vector [145]	75
Table 4-6: Attributes and their value types [145]	77
Table 4-7: Normal and attack data in different types of NSL-KDD dataset [145]	78
Table 4-8: Network protocols used by various attacks [145].....	79
Table 4-9: Attack sub-classes of UNSW-NB 15 data	83
Table 4-10: The flow features of UNSW-NB 15 data [148].....	85
Table 4-11: The Basic features of UNSW-NB 15 Data [148]	86
Table 4-12: The content features of UNSW-NB 15 data [148]	87
Table 4-13: The Time features of UNSW-NB 15 Data [148].....	88
Table 4-14: The Additional Generated features of UNSW-NB 15 [148]	89
Table 4-15: Different data type features of UNSW-NB 15 Data.....	91
Table 5-1: Representation of non-numerical characters to numeric in NSL-KDD dataset ...	100
Table 5-2: Representation of non-numerical characters to numeric in UNSW-NB-15 dataset	101
Table 5-3: Selected feature for injection attack identification	113
Table 5-4: List of transfer functions commonly used in deep learning	124
Table 5-5: Different combinations of hybrid deep learning.....	142
Table 5-6: Attack class classification NSL-KDD dataset	144
Table 5-7: Attack class classification UNSW-NB-15 dataset.....	145
Table 6-1: Classification accuracies of deep learning algorithms on NSL-KDD dataset	147
Table 6-2: Benachmarking detection accuracies of propped CNN algorithm on NSL-KDD dataset.....	152
Table 6-3: Classification accuracies of deep learning algorithms on measurement injection attacks.....	153

Table 6-4: Classification accuracies of deep learning algorithms on command injection attacks.....	154
Table 6-5: : Classification accuracies of different hybrid deep learning algorithms on NSL-KDD dataset	155
Table 6-6: Training and Fine-tunings times of different deep learning algorithms on different CPU's.....	156
Table 6-7: Training of deep learning algorithms on different hardware accelration platforms in serial and parallel modes	157
Table 6-8: Recommendation of the deep learning algorithm in realtion to the input parameters	159
Table 6-9: Recommendation of the deep learning algorithm in realtion to the input data types	160
Table C-0-1: Evaluation of performance parameters for identification of injection attacks in automation applications.....	258
Table D-0-1: Planned attack classes and their types	259
Table D-0-2: Features of the generated dataset.....	265

List of Acronyms

<i>2D</i>	Two Dimension
<i>AE</i>	Autoencoder
<i>ANN</i>	Artificial Neural Network
<i>CD</i>	Contrastive Divergence
<i>CIA</i>	Confidentiality, Integrity, Availability
<i>CMRI</i>	Complex Malicious Response Injection
<i>CNN</i>	Convolutional Neural Network
<i>CPS</i>	Cyber Physical Systems
<i>CPU</i>	Central Processing Unit
<i>DBN</i>	Deep Belief Networks
<i>DCS</i>	Distributed Control System
<i>DDoS</i>	Distributed Denial of Service
<i>DNN</i>	Deep Neural Network
<i>DoS</i>	Denial of Service
<i>DT</i>	Decision Tree
<i>FDIA</i>	False Data Injection Attacks
<i>FN</i>	False Negatives
<i>FP</i>	False Positives
<i>GPU</i>	Graphical Processing Unit
<i>HMI</i>	Human Machine Interface
<i>ICS</i>	Industrial Control System
<i>IDS</i>	Intrusion Detection System

<i>IP</i>	Internet Protocol
<i>IT</i>	Information Technology
<i>KDD</i>	Knowledge Discovery and Datamining
<i>KNN</i>	K-Nearest Neighbour
<i>MFCI</i>	Malicious Function Code Injection
<i>ML</i>	Machine Learning
<i>MPCI</i>	Malicious Parameter Command Injection
<i>MSCI</i>	Malicious State Command Injection
<i>NMRI</i>	Naive Malicious Response Injection
<i>NN</i>	Neural Network
<i>PCA</i>	Principal Component Analysis
<i>PCS</i>	Process Control Systems
<i>PLC</i>	Programmable Logic Controller
<i>R2L</i>	Remote to Local
<i>RBF</i>	Radial Basis Function
<i>RBM</i>	Restricted Boltzmann Machines
<i>ReLU</i>	Rectified Linear Unit
<i>RNN</i>	Recurrent Neural Network
<i>ROC</i>	Receiver Operating Curve
<i>RTU</i>	Remote Telemetry Units
<i>SAE</i>	Stacked Autoencoders
<i>SCADA</i>	Supervisory Control and Data Acquisition
<i>SGD</i>	Stochastic Gradient Descent
<i>SMR</i>	SoftMax Regression

<i>SVM</i>	Support Vector Machines
<i>TCP</i>	Transmission Control Protocol
<i>TEP</i>	Tennessee Eastman Process
<i>TN</i>	True Negatives
<i>TP</i>	True Positives
<i>U2R</i>	User to Root
<i>VPN</i>	Virtual Private Network

List of Publications

The following list of papers have been published, presented and contains material based on the content of this thesis.

Conferences:

- I. Sasanka Potluri and Christian Diedrich, "High Performance Intrusion Detection and Prevention Systems: A Survey," *15th European Conference on Cyber Warfare and Security*, ECCWS 2016, pp.260, 2016. ACPIL ISBN: 978-1-910810-93-4
- II. Sasanka Potluri and Christian Diedrich, "Accelerated deep neural networks for enhanced Intrusion Detection System" *21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016*, Berlin, 2016, pp. 1-8. doi: 10.1109/ETFA.2016.7733515
- III. Sasanka Potluri, Navin Francis Henry and Christian Diedrich, "Evaluation of hybrid deep learning techniques for ensuring security in networked control systems," *22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2017*, Limassol, Cyprus, 2017, pp. 1-8. doi: 10.1109/ETFA.2017.8247662
- IV. Sasanka Potluri, Christian Diedrich and Girish Kumar Reddy Sangala, "Identifying false data injection attacks in industrial control systems using artificial neural networks," *22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2017*, Limassol, Cyprus, 2017, pp. 1-8. doi: 10.1109/ETFA.2017.8247663
- V. Sasanka Potluri, Shamim Ahmed and Christian Diedrich, "Convolutional Neural Networks for Multi-Class Intrusion Detection System," *6th International Conference on Mining Intelligence and Knowledge Exploration, MIKE 2018*, doi: 10.1007/978-3-030-05918-7_20
- VI. Sasanka Potluri, Sai Ram Rao Nanduru, Kishore Vasamshetty, Christian Diedrich, "Development of Injection Attacks Toolbox in MATLAB/Simulink for Attacks Simulation in Industrial Control System Applications" *17th IEEE International Conference on Industrial Informatics. INDIN 2019*, Helsinki-Espoo, Finland
- VII. Sasanka Potluri, Christian Diedrich, "Deep learning based efficient anomaly detection for securing process control systems against injection attacks," *15th IEEE International Conference on Automation Science and Engineering, CASE 2019*, Vancouver, Canada, 2019

Journal:

- VIII. Sasanka Potluri and Christian Diedrich, "Deep Feature Extraction for multi-Class Intrusion Detection in Industrial Control Systems," *International Journal of Computer Theory and Engineering* vol. 9, no.5, pp. 374-379, 2017. doi: 10.7763/IJCTE.2017.V9.1169

Book Chapter:

- IX. Sasanka Potluri, Shamim Ahmed and Christian Diedrich, “Securing Industrial Control Systems from False Data Injection Attacks with Convolutional Neural Networks,” *Development and Analysis of Deep Learning Architectures*, Chapter 8, Springer, 2020, doi: 10.1007/978-3-030-31764-5_8

Intentionally left blank

1. Introduction

Cyber-attacks through internet have proliferated in recent years. Hence, security is considered as a critical concern in many domains and Industrial Control System (ICS) is not an exemption. This chapter gives a brief introduction to ICS architecture, followed by the importance of developing an exclusive security concept for ICS in contrast to the available Information Technology (IT) solutions. A discussion on different types of attacks on ICS is done along with some famous example for attacks on ICS.

1.1. Background and Motivation

ICS refers to a variety of systems comprised of computers, electrical and mechanical devices and manual process overseen by humans. They perform automated or partially automated control of equipment and are an integral part of several infrastructures. Main components are services which include Supervisory Control and Data Acquisition (SCADA), Distributed Control Systems (DCS), Programmable Logic Controllers (PLC) and devices such as Remote Telemetry Units (RTU), smart measurement/actuation devices and intelligent field instruments including remotely controlled devices and intelligent electronic relays [2]. A typical architecture of an ICS is shown in Fig.1 below [3].

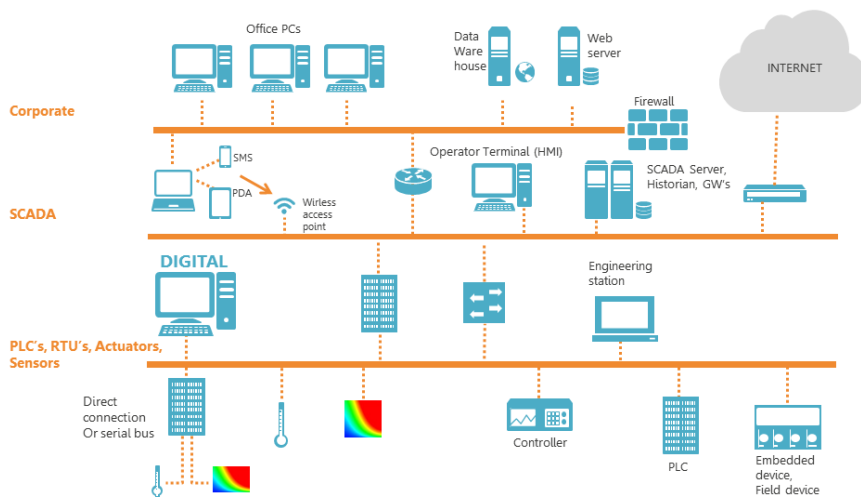


Figure 1: Typical ICS architecture

Among others, the primary function of an ICS includes collecting data from physical processes and sending commands to control these physical processes, thereby creating a feedback control

loop. ICS components are networked [4], still many of the industries pre-assume some of the following myths [5]

- The plant is not connected to the internet
- The plant is secure because it has a firewall
- Hackers do not understand SCADA/ DCS/ PLC
- The industries are not a target
- Available safety systems will prevent any harm

There are different versions of these myths and many other but still the summary is the same; ICS is not secure. From the architecture shown in Fig.1. it is clear that the attack on ICS can be possible at different levels ranging from low level field devices (PLC's, RTU's, Actuators, etc.) to a high level SACADA/corporate network. The ICS must ensure three of the following security attributes/qualities famously know as CIA (Confidentiality, Integrity, Availability) triad [6] shown in Fig. 2.

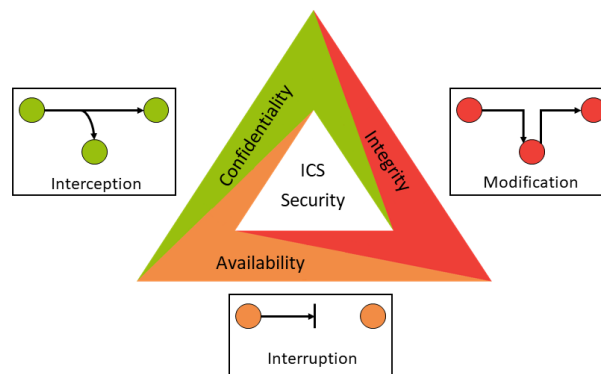


Figure 2: CIA triad and causes for their violations

- **Confidentiality:** Is the property, that information is not made available or disclosed to unauthorized individuals, entities or processes. Interception of data leads to a violation of confidentiality.
- **Integrity:** Is the property of maintaining and assuring the accuracy and completeness of data over its entire life-cycle or protecting information from being modified by unauthorized parties. External modification of the data leads to a violation of integrity.
- **Availability:** Is the property to ensure that authorized parties or entities able to access the information or devices when needed. Interruption of data arrival or transfer leads to violation of availability.

If any of these security attributes are not satisfied, the system is considered to be under threat or attacked. ICS differ quite significantly from traditional enterprise networks due to the specific requirements of their operations especially reliability and safety. Hence the priority of the security attributes also differs in each case. The following Table 1-1 provides the differences in priority of security attributes of an IT to ICS infrastructure.

Table 1-1: Priorities of security attributes in IT vs ICS

Priority	Information Technology (IT)	Industrial Control Systems (ICS)
1	Confidentiality	Availability
2	Integrity	Integrity
3	Availability	Confidentiality

Existing security approaches such as firewalls, cryptographic primitives etc. are either inapplicable, insufficiently scalable, incompatible or inadequate to secure the ICS. While strong concerns about the security of ICS, particularly in the context of critical national infrastructure, were expressed even in the early 2000s [7]. To address this issue, a large number of defences against attacks have been proposed in the literature. Despite all the efforts made by researchers in the community over the decades, the security problem is not solved and always brings new challenges. One prominent reason for that could be rapid growth in computational power and available resources to attackers, which enable them to launch complex attacks [8]. [9] provides a real-time threat map, that how the attacks packets were published from one region to other and it is surprising to see that more than 95% of those attack patterns are unknown. Some of the known ICS cyber-attacks are:

- **Trojan attack:** In 1982, it was the first know attacks on the critical infrastructure occurred in Serbia. Trojan was used to insert malicious values in pump speeds and valve settings which created high pressure beyond acceptable to the pipeline joints and welds resulted in an explosion. This attack can be considered as an injection attack on ICS [10].

- ***SQLSlammer***: In 2003, the SQLSlammer worm infected a Supervisory Control and Data Acquisition (SCADA) system that controlled the Davis-Besse nuclear plant in Ohio. The worm shutdown the HMI of the supervisor SCADA systems that handled the plant's safety systems. This can be considered as Denial of Service (DoS) attack on ICS [11].
- ***Operation Ghoul***: In August 2016, Kaspersky Labs unearthed a spear phishing campaign that was targeting industrial organizations. The attack started with an email that appeared to be coming from a bank in UAE. This email is attached with a malware named HawkEye, which collects the personal information through key strokes and clipboard data. Around 130 organizations across the globe were impacted by this attack. This attack can be considered as a probing attack on ICS [12].
- ***New York Dam attack***: U.S. Infrastructure online was attacked and infiltrated the computerized controls of a New York Dam. Justice department claimed it that it was done by Iranian hacker. The attackers broke into the command and control system of the dam in 2013, through a cellular modem. Even though the attack happened in 2013, it was only in 2016 that the cyber-attack was affirmed [13].

The mentioned attacks are just a few but a lot of attacks still exists and every day new attacks are being identified. Therefore, novel technologies need to be adopted for the development and to secure ICS from these attacks.

In general, defence against attacks consists of preparation, detection and reaction phases. A risk analysis is usually conducted by security engineers during the preparation phase to understand the environment and the assets they are trying to protect in that environment. This process is very crucial because it helps the engineers to understand how attacks can take place and how they affect the network [14]. The preparation phase also includes identification of infrastructure vulnerabilities, development of security strategies and plans and installation of required security devices based upon analysis of the information gathered [15] [16]. Another key element of the security concept is the detection system. An Intrusion Detection System (IDS) usually complements a firewall to forma an effective cyber security solution. One security motto is "Prevention is ideal but detection is a must" [17]. Efficient detection of attacks is required to able to react properly. Thus, a detection phase is of paramount importance in development of defence-in-depth concept. Finally, handling detected intrusions is done during the reaction phase. For this purpose, it is also vital to know the type of attack. As a consequence, this thesis

prime concentration is on securing the ICS infrastructure from possible attacks by appropriate design and use of IDS through novel deep learning algorithms.

In general, the process of intrusion detection is performed using two basic approaches. The first one is signature-based detection also termed as misuse detection [18]. In this type, IDS searches for the evidence of attacks based on the knowledge accumulated from the known attacks. This knowledge is represented by attack signatures which are patterns or sets of rules that can uniquely identify an attack. Based on the knowledge of the past intrusions or known vulnerabilities, these signatures were designed. The key advantage of this approach is good detection accuracy on the contrary, the main drawback is its inability to detect novel attacks. The second type of IDS is anomaly-based intrusion detection, also termed as behaviour-based detection [19]. This approach builds models of the legitimate activities and any deviations from these legitimate actions are termed as anomalous or attacks. The key advantage of this approach is its ability to detect novel or unforeseen attacks on contrary the main drawback is its high false alarm rate. A more detailed discussion of the type of detection mechanisms is done in Chapter 3.

In order to cover these existing challenges and secure the ICS from such attacks, we found it is necessary to improve the detection accuracy of the anomaly-based intrusion detection using novel deep learning algorithms.

1.2. Problem statement

Existing cybersecurity mechanisms either suitable for IT infrastructure or use misuse detection techniques. The knowledge and patterns of previous attacks are used to create them as signatures which accurately identify new instances of those attacks. But these techniques are not suitable for identifying novel attacks as there is a need to update the signatures after identifying the attacks, which is not so fast and not feasible every time. Novel attack detection through anomaly detection approaches is possible at the expense of falsely identifying novel activity as malicious. Even with the modern infrastructure, existing approaches are unable to handle the bandwidth of data and perform the attack detection mechanism efficiently. Despite detection, in order to respond to the identified attacks, it is necessary to know the attack type. Many existing approaches fail or inaccurate in identifying the attack classes which close the door to perform prevention mechanism. These limitations lead to a significant number of attacks being missed or misclassified paving the way to successful attacks on ICS.

Machine learning techniques have the potential to overcome some limitations of intrusion detection systems as discussed in (Chapter 3) and their drawbacks were identified. The main ***problem*** is that existing machine learning techniques cannot extract the necessary features from the network packets that are necessary for identification of the complex nature of the novel attacks. Similarly, normal feature extraction techniques such as statistical features and mathematical features are also not sufficient in identifying attacks that are intelligently concealed in the plant data. So, extracting such complex correlations from the input data also termed as features is therefore a critical step which influences the attack detection accuracy. However, extracting such complex features is often an ad-hoc process, using trial and error to find which identified features are more relevant for the detection processes. Such a process requires domain knowledge, and it is time consuming when done iteratively. Other difficulties specific to the field of ICS security include:

- A huge volume of data is generated
- Malicious data is in general only a tiny fraction of the total traffic
- Accurately labelled training data is difficult to obtain
- ICS has a high diversity
- The detection algorithms are expected to have low false detection rates

These difficulties in intrusion detection motivate the application of deep learning techniques for extracting complex features in cybersecurity for ICS. In this thesis, an investigation is done on how to construct these features from the obtained data to perform multi-class classification.

1.3. Research aims

To address the problem statement, we have the following research aims.

Aim 1: Investigating existing applications and approaches of different machine learning techniques to ensure the security of ICS, concentrating on feature extraction and multi-class attack classification.

Since the correlation between the input dataset is a key aspect in identifying the attacks, feature extraction has been identified as a key issue in intrusion detection. We explore different existing techniques presented in the literature. The literature review focuses on both network intrusion as well as injection attack detection.

Aim 2: Designing different deep learning techniques which can be used to find relevant features from network traffic for given network-based intrusion detection in ICS.

To overcome the existing drawback of misuse-based intrusion detection and machine learning based anomaly detection techniques, we propose the use of deep learning. Intrusion detection using deep learning can learn from previous attacks and investigations. They can interpret the complex correlations between the input data and extract the necessary features for proper classification. They can detect novel attacks as either similar to know attacks or as anomalies. They are even capable of processing large volumes of data in an automated way for fast feature extraction and detection. As it is complex to identify whether the extracted features are good or not, a proper feature engineering is developed so that the deep learning algorithms can identify the relevant features.

Key or relevant features enable the deep learning-based IDS to discriminate normal and malicious behaviours. However, if the deep learning algorithms are provided with some irrelevant features, it may not be able to identify the attacks. Hence, proper training of the algorithm to extract the relevant features is a key point for the classifier's performance. In general, proper feature identification requires a huge domain knowledge, but the aim is to identify an automated features extraction process based on the relevant input features.

The aim is to identify which deep learning algorithms were able to identify the propose features for detecting the attack patterns efficiently. The research also focuses on identifying which deep learning algorithm are able to identify which types of attacks efficiently.

Aim 3: Using those complex features generated through Aim 2, build an appropriate classifier for multi-class classification in identifying network attacks and test their effectiveness.

The multi-class classification of network attacks is achieved through the use of key features from the network traffic dataset. The features are used for training the classifier such as SoftMax or support vector machines. In particular, the aim is to detect all attack classes with maximum possible accuracy.

Aim 4: Identification of proper feature extraction and classification mechanism for the detection of injection attacks on ICS.

Injection attacks are considered to be hard to identify attack types in the scope of ICS cybersecurity. They mitigate all possible network IDS strategies and modify the datagrams of the network packets. From Aim 2, proper feature extraction strategies need to be taken and their

correlation with traditional features extraction techniques needs to be evaluated. Later, the features are used for training classifier to identify the normal and attack patterns in the plant data.

Aim 5: Development of an injection attack toolbox for the future generation of multiple injection attack types dataset in a simulation environment.

As there doesn't exist a standard dataset for testing and evaluating the performance of the IDS against injection attacks and each ICS application is independent, simulation of an injection dataset is of key importance. This simulation toolbox will pave a way in generating the injection attack dataset for different applications based on their needs.

1.4. Research contributions

The main goal of this work is to extract the complex features out of the network packets as well as from the datagrams of the network packets to identify different attack classes along with recommending appropriate security mechanism techniques for protecting ICS against cyber-attacks. As a part of achieving this goal, the thesis contributes to carry out different tasks to achieve the aims mentioned above. The contributions of this thesis are categorized into the following:

Contribution 1: Development of a deep learning-based features extraction mechanism to generate a set of complex features from raw network data. This work is described in Section 7.1.

Use of deep learning algorithms such as Stacked-Autoencoders (SAE), Deep Belief Networks (DBN) and Convolutional Neural Network (CNN) for extracting proper feature that incorporates the relevant correlations in the input data of the network traffic. These extracted features are further used for the development of multi-class attack classification.

Contribution 2: Development of Multi-class attack classification using a combination of deep learning-based feature extraction with machine learning algorithms such as SoftMax and SVM.

This contribution addresses Aim 3. Different classification algorithms were considered for training with the features extracted through contribution 1. In order to evaluate the efficiency of the features along with the classification algorithms, performance metrics mentioned in Section 3.5. were calculated and compared with the different combinations of feature extraction

mechanisms and classification algorithms. The best configuration parameters and combinations were presented with their critical advantages.

Contribution 3: Combination of traditional features extraction techniques with the deep learning-based feature extraction mechanisms for the identification of injection attacks in the datagrams of the network packets.

To accomplish Aim 4, the traditionally extracted features are given to a deep learning-based features extraction mechanism such as SAE, DBN and CNN for the extraction of complex features. These features are used for training the classifiers for efficient identification of normal and attack packets. Performance metrics are calculated and evaluated to identify the feasibility of the algorithms.

Contribution 4: Development of an injection attack toolbox in MATLAB/Simulink as a toolkit to use with different ICS simulation environments to develop the defence-in-depth strategies in industries.

In order to fulfil the Aim 5, an injection attack toolbox is developed in MATLAB/Simulink. The different classes of injection attacks are developed in accordance with the attack types mentioned in [20]. This toolbox is very versatile and can be configured in such a way that it can be used with all simulated ICS models and configure the key parameters according to the individual needs.

1.5. Outline of the document

The rest of the thesis is organized as follows: Chapter 2 gives detailed information about the possible types of attacks on ICS. The two categories of attacks; network attacks, injection attacks and their types are discussed in detail. How these attacks stipulate the key security parameters (CIA) is also explained.

Chapter 3 provides a comprehensive review on the complete relevant literature related to the taxonomy of the intrusion to the different intrusion detection systems followed by the summary of most used and commonly mentioned techniques for the development of IDS and their drawbacks are discussed. It also discusses the basic evaluation metrics that are used to evaluate the efficiency of the developed algorithms.

Chapter 4 offer a detailed information on the datasets used in part of the scope of the research in detail. Two datasets, namely NSL-KDD and UNSW15 were used for the evaluation of

intrusion detection capabilities for network attacks and proceed control plant data for the detection of injection attacks is discussed. The description of the datasets includes the types of attacks and their fraction to the normal data is included.

Chapter 5 discuss the step by step approach of the proposed security schema using deep learning algorithms. The basics concepts along with their usage in the proposed approach, are discussed. A step by step procedure for the implementation of deep learning algorithms for the securing ICS is discussed. The implementations of the deep learning algorithms with the datasets mentioned in chapter 4 to achieve the aims that are discussed earlier in section 1.3. It also considers the use of hybrid deep learning algorithms that is a combination of different machine learning and deep learning algorithms as different classifier combinations to improve the accuracy in multi-class attack classification. The importance of using high-performance computing is also discussed.

Chapter 6 presents the selected results of this thesis, followed by a discussion and evaluation based on the obtained results. Some recommendations on the choice of algorithms are made based on the acquired results.

Chapter 7 from the outcome after evaluation, it is found that in order to develop an IDS against injection attacks, a dataset with possible attacks is necessary. Due to this reason, an injection attack toolbox is developed and described in this chapter for other researchers to use it.

Finally, the thesis is concluded in Chapter 8, which summarizes the entire work and presents some suggestions for further research in the area of intrusion detection.

2. Classification of cyber-attacks on ICS

"ICS are widely used in many industries like chemical processing, petroleum refining, electrical power generation and distribution, water purification and distribution, intelligent buildings and nuclear plants" [21]. Their primary functions include collecting data from remote physical processes and sending commands to control these physical processes, thereby creating a feedback control loop. There have been several infrastructure vulnerabilities with the ICS infrastructures and hence these systems are prone to intrusions and cyber-attacks. These attacks are so critical enough to cause a variety of financial damage and harmful effects on humans and surroundings [22]. The common attacks that are targeted on an ICS can be grouped under any of these following two classes as seen in Fig. 3.

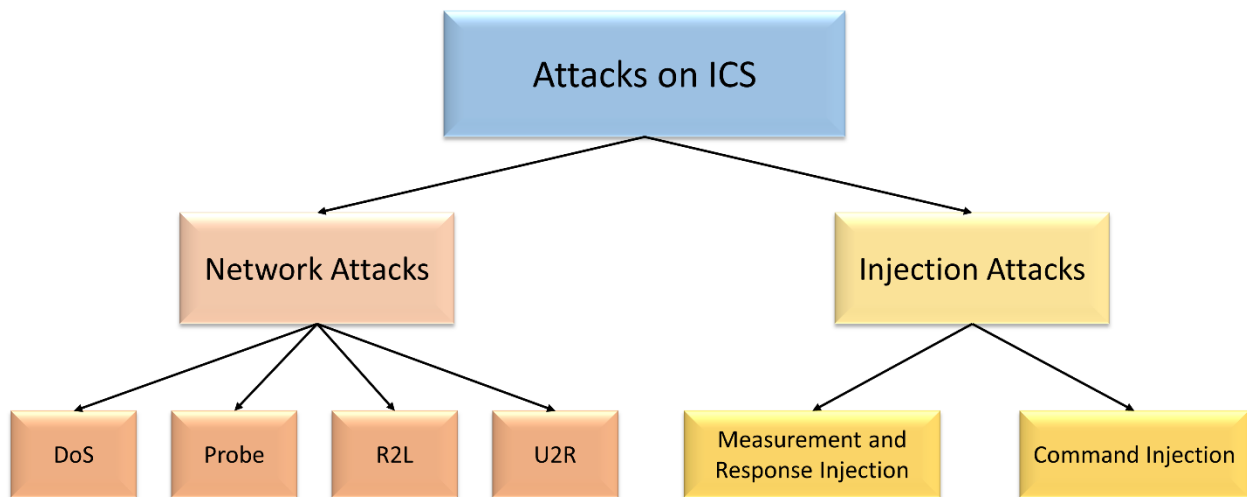


Figure 3 : Classification of possible cyber-attacks on ICS

2.1. Network attacks

The attacks types can change significantly from one year to the next. Many of the actual attacks involve combinations of vulnerabilities. Some common attacks on ICS can be:

Denial of Service (DoS)

"An attack against ICS to stop the proper functioning of some portion of an ICS or to effectively disable the entire system. These attacks can target the connected physical system or the ICS itself. DoS against physical system vary from opening or closing of valves manually and switching to the destruction of portions of the physical process that prevent operation. DoS against the ICS target the communication links or attempt to disable programs running on system endpoints which control the system, log data and govern communications [23]. DoS

attack mainly affects '**availability**' which is the main priority in ICS. Some examples of DoS attacks are apache, smurf, Neptune, ping of death, back, mail bomb, UDP storm etc" [24].

Probe

"It is an action taken or an object used for the purpose of learning something about the state of the network. It collects or monitors network activity and attempt to gain access to a computer and its files. Probe attack mainly effects the '**confidentiality**' in ICS. This technique is commonly used in data mining. Some examples of probe attacks are saint, portsweep, mscan, nmap etc" [24].

Remote to Local (R2L)

An R2L is an attack in which a user sends packets to a machine over the internet, which s/he does not have access to in order to expose the machines vulnerabilities and exploit privileges which a local user would have on the computer. Some examples of R2L attacks are xlock, guest, xnsnoop, phf, sendmail dictionary etc.

User to Root (U2R)

U2R attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain superuser privileges. Some examples of U2R attacks are perl, xterm etc.

A more detailed information about different types of network attacks and their detection mechanisms can be found in [25]. The mentioned attacks present here are taken from the famous benchmark dataset NSL-KDD. These are just some categories of attack classes possible in network attacks. The other possible attack types from another dataset are discussed in Chapter 4.

2.2. Injection attacks

"Injection attacks are also known as False Data Injection Attacks (FDIA) where the attacker gains access to a critical process or process parameters in ICS and forces the system to execute newly introduced code or command. Injection attacks effect the '**integrity**' of an ICS" [24]. FDIA are broadly classified into three categories. Common injection attack categories are:

- **Response Injection:** "ICS protocols often take the first response packet to a query and reject subsequent responses as erroneous. This enables to craft response packets and use

timing attacks to inject the responses into a network when they are expected by a client" [26].

- **Measurement Injection:** "Falsified process measurements are injected into the ICS in this type of attacks. The attacker simulates a process measurement such as a water level or gas pressure increasing or decreasing, which in turn generate false actuations" [26].
- **Command Injection:** "False control or configuration commands are injected into the ICS in this type of attacks. Potential impacts of this attack include interruption of process control, interruption of device communications, unauthorized modification of device configurations etc" [26].

Apart from the key attack classes, there are several subclasses in these attack types. The different types of attack classes and their categories are mentioned in Table 2-1. This table provides info about the name of the attack their classification category and the level in ICS where the attack takes place in relation to the behaviour and system. These attack classes are the sub-classes of the injection attack classes mentioned in Fig. 3.

Table 2-1: List of FDIA attacks and their sub-classes [20]

S.No	Attack Name	Classification	Level
1	Naïve Read Payload Injection	NMRI	2
2	Invalid Read Payload Size	NMRI	2
3	Naïve False Error Response	NMRI	2
4	Sporadic Sensor Measurement Injection Attack	NMRI	1
5	Slope Sensor Measurement Injection	CMRI	0
6	High Slope Measurement Injection	CMRI	0
7	High Frequency Measurement Injection	CMRI	0

Classification of cyber-attacks on ICS

8	Altered System Control Scheme	MSCI	1
9	Altered Actuator State	MSCI	1
10	Altered Control Set Point	MPCI	3
11	Force Listen Only Mode	MFCI	3
12	Restart Communication	MFCI	3

2.2.1. Naive malicious response injection (NMRI)

In this type of attack, the attacker will successfully inject response packets into the network but lacks information about the process that is being monitored and controlled. The attacker will inject invalid payloads in the form of all zeroes, negative numbers, large numbers, etc. This attack lacks sophistication and causes an alarm. There are 4 types of NMRI and they include:

1. *Naive read payload size*: The first form of NMRI attack which is based on network protocol knowledge. Read coil, discrete input, holding register and input register queries will have a quantity field to specify the number of objects returned by the server. An attack is injected such that the server will respond with the correct number of objects on request, but with random contents.
2. *Invalid read payload size*: This is another type of NMRI attack, where the requested number of objects from the read coil, discrete input, holding register or input register queries is ignored. The response payload will be either larger or smaller than the requested amount.
3. *Naive false error response*: It is an NMRI attack, where false error messages are sent to a client on a read command.
4. *Sporadic sensor measurement injection*: A kind of NMRI attack, that sends sporadic false process measurements outside the bounds of the high (H) and low (L) control set points. But these false measurements are not outside the alarm set point range.

2.2.2. Complex malicious response injection (CMRI)

CMRI attacks are very sophisticated compared to NMRI. The attacker would have a good understanding of the ICS architecture that is to be attacked. This attack will mask the real state

of the physical process being controlled, thereby affecting the feedback control loop managing the cyberphysical system in a negative manner. There can be different forms in which a CMRI attack can be injected and they include:

1. *Calculated sensor measurement injection:* This is a type of CMRI attack where calculated process measurements are injected. It requires system knowledge and an accurate model of the system to be attacked. For example, an attacker can send false response packets to indicate that water level is full to turn off the water pump while the actual water level is much below.
2. *Replayed measurement injection:* Here, the attacker replays the captured process measurements to the client. This gives an impression to the operator that the system is running normally.
3. *High frequency measurement injection:* A special type of calculated sensor measurement injection where frequency of process measurement changes is increased beyond the normal rate. The attack scenario may appear similar to the system behaviour at a different time of a day and cause the operator to misconfigure the system to handle the falsified demand.

2.2.3. Malicious state command injection (MSCI)

In MSCI attack type, the state of the process control system will be changed from a safe state to critical state by sending malicious commands to remote field devices. The attack can be planned with single or multiple injected commands. Different types of MSCI are listed as follows:

1. *Altered system control scheme:* Most of the control systems allow operators to change the control modes between automatic and manual. In this type of MSCI attack, these control modes are changed. For example, consider a gas pipeline control system. During automatic mode, the pump is controlled from a PLC and in the manual mode PLC is not used to control. MODBUS commands are used to set or change values stored for system control mode, pump state and solenoid state. An attacker can inject commands to change the control mode from automatic to manual and this, in turn, can be very critical if the gas pressure exceeds certain limit.
2. *Altered actuator state:* In this attack type, the command injection is used to change the system actuator states for one time. In a gas pipeline system, the pump can be turned ON or OFF and the relief valve can be made open or close.

2.2.4. Malicious parameter command injection (MPCI)

Altered control setpoint is an MPCI attack type. The device set points are changed in this type of attack. The setpoints are used to provide variable control on a system. In a water tank system, the command injection can be used to alter the high and low setpoints. This attack can also be used to change the alarm values stored in PLC registers. To disable the alarms, values are altered to different high and low setpoints.

2.2.5. Malicious function code injection (MFCI)

MFCI can be classified into 2 types:

1. *Force listen only mode:* This attack type will cause the server not to respond to queries and will no longer transmit on the network. Most of the industrial control systems use the polling technique, where master node like human machine interface (HMI) will frequently poll the servers for data. The data displayed on HMI is used by human operators to take supervisory control. Using this type of command injection, the attacker will place the server (MODBUS) in listen only mode and the server is prevented from responding to any queries. Hence there is a loss of system visibility and control.
2. *Restart communication:* In this attack, the attacker sends a command to restart the server, which leads to temporary loss of communication. This leads to an inability to observe and control the process. Multiple successive restart communication attacks can lead to complete loss of communication between the process and control.

These attack types and attack classes defined the number of possibilities of the possible attacks on an ICS infrastructure. A more detailed informant about the theory of injection attacks can be found at [27] and some important points are discussed in Chapter 3. In this thesis, research efforts are made in the identification of some of those attacks accurately, but further research is necessary in future to develop the defence mechanisms against all possible attacks.

3. Literature review

This chapter provides a detailed overview on the background of the attack on ICS, a detailed overview on IDS and the available research work on the development of IDS using machine learning and deep learning techniques on network and injection attacks.

3.1. Background

"In 1990's, ICS cybersecurity awareness was very low and its perceived importance is even lower. Generally, it was viewed as a corporate Information Technology (IT) issue with little direct impact on powerplant or grid operation. Moreover, it was viewed as a hindrance to ICS technology advancements. From a security perspective, ICS were generally isolated networks and the concept of "security by obscurity" was alive and well. As security was not a consideration, there was little reason to question the need for tighter system integration" [24].

Existing security approaches such as firewalls, cryptographic primitives etc. are either inapplicable, insufficiently scalable, incompatible or inadequate to secure the ICS. Recent events have shown that cyber-attacks on industrial control systems are becoming increasingly sophisticated. Cyber-attacks with the ability to compromise physical equipment are considered as the most trivial forms of attacks [28]. Disabling or tampering with physical equipment can easily render them unavailable at critical times of operations, while operational reliability is of the utmost importance in smart grids. Threats are evolving over time, while cybercriminals are becoming smarter and smarter, less so their victims. Cyber-attacks in the past were generally one-dimensional and mainly in the form of denial of service (DoS) attacks, computer viruses or worms, or Trojan horses. However, this has fundamentally changed in recent times. Cyber threats are undergoing a diversification that is resulting in the combination of the "Internet", "teamwork" and "commercial interests", while appearing in multiple forms [29]. BlackEnergy malware is one example of such threats, which has evolved over time from a simple distributed denial of service (DDoS) platform to rather sophisticated plug-in based malware [30]. Moreover, BlackEnergy has been used in numerous targeted attacks [31] since its discovery in 2007.

By exploiting vulnerabilities, an attacker can infect systems with malware, propagate malware within the system (or even between different systems) and use additional attack methods to achieve his/her ultimate goal. In this regard, as a single act of penetration is often not sufficient this leads to a situation involving multistage attacks, which are composed of a number of

dynamically interrelated attack steps, where the occurrence of the next step depends on the successful completion of the previous step. The Stuxnet cyberattack on the Iranian nuclear programme is the best-known example of a multistage attack on physical infrastructure [10]. Stuxnet infected approximately 100,000 hosts across over 155 countries prior to September, 2010, according to the Symantec report [11]. More recent, widely known multistage cyberattack scenarios include the German steel mill breach in December 2014 [12] and the Ukrainian electric power disruption in December 2015, which will be briefly described in the following paragraphs:

In present days scenarios, the modern ICS is not secure anymore and are under continuous attacks. The cyber defences of the ICS are being tested, probed and targeted at an unprecedented rate by a variety of attackers with different motives, skill levels and techniques. The concept of Industry 4.0, a combination of CPS and Internet of Things (IoT) on one side brings the features of interoperability and flexibility but on other side exposes to the risk of more cyber-attacks. Attacks on ICS are not new, but the recent trend of increasing attacks on ICS will highlight the necessity of a robust mechanism for ensuring the security of ICS [32].

3.2. Intrusion detection system

Attacks on industrial systems are performed by threat actors with varying sophistication and goals. While it is not possible to list all the potential attacks, it is good to know that the list is always growing. Some types of attacks on communication network are vulnerabilities [33][34], SYN flooding [35], Distributed Denial-of-Service (DDOs) [36] [37], surfing and the list goes on. Intrusion detection refers to the detection of malicious activity (attacks, break-ins, penetrations and other forms of computer abuse) in a computer related system or in the communication networks. An intrusion can be sometimes identified as a completely different behaviour from the normal and sometimes hard to identify it from normal behaviour. These malicious activities or intrusions are very important in network security perspective. Due to these complexities, there doesn't exist a unique technic that can identify all types of attacks or intrusions.

As IDS deals with only detection, it is considered as a passive mechanism only. In order to prevent attacks, we need systems that can detect attacks online and prevent malicious data from entering the network. Hence IDS is improvised as IPS (Intrusion Prevention System) in some literature this is also termed as IDPS (Intrusion Detection and Prevention Systems).

In the early stages, a system administrator monitored the systems personally and identify the unusual activities in the network [38]. Later, some improvements occurred in the late 70's where the administrators audited the logs by printing the log files and searching them for a suspicious activity [39]. This approach is time consuming and easy to miss such suspicious activities. After the storage technology became less expensive, the logs were moved to online and programs and technologies were developed to analyse the data. Despite these technologies are slow and computationally intensive, they somehow match with those days network requirements [40]. The steps for the development of modern days IDS was developed in the early 90's [41]. Since then, the effectiveness and efficiency of the IDS is based on how fast the system could identify an intrusion by maximizing the True Positives and minimizing the False Positives [42].

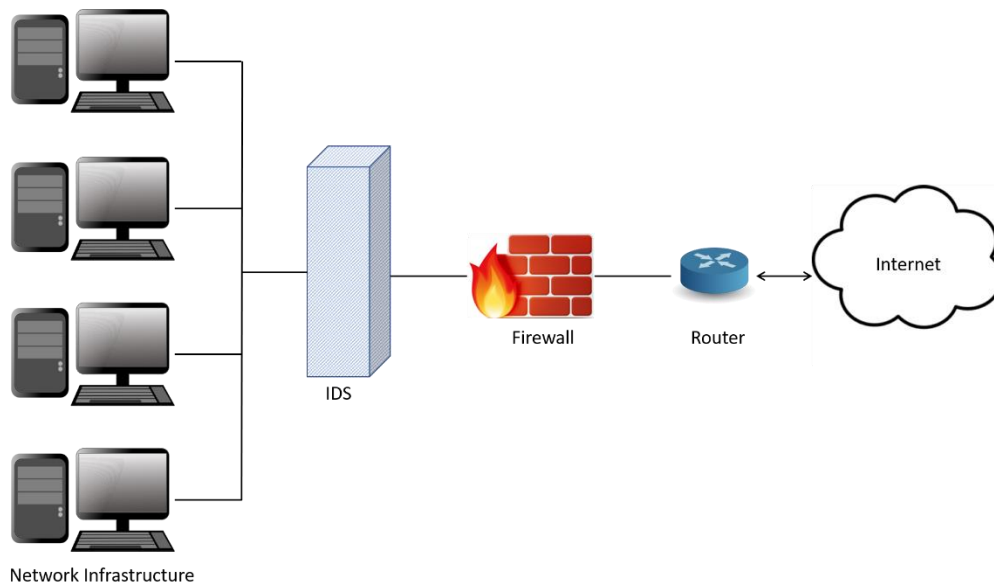


Figure 4: Possible placement of an IDS in a network infrastructure

An IDS is a software or hardware that helps to protect from and ward off attacks and penetration attempts to the network. The key difference from an IDS to a simple firewall or adaptive proxy firewall is that the firewalls can block connections. IDS is a combination of several individual intrusion detection techniques available (signature analysis, traffic monitoring and anomaly detection). IDS check the network behaviour and find the nodes that are not working normally. It is an additional unit installed at the clients or server or both [43] [44] . The typical placement of an IDS in a network is depicted in Fig. 4.

From Fig. 4, it is clear that the IDS is placed after the firewall, the typical protective mechanism to any network. IDS guard's the network infrastructure from the attacks that overcome through

the traditional firewall protection. So, it lies between the network infrastructure and state of the art protection mechanisms.

Several characteristics are needed in the development of an IDS to have optimized performance, maximizing the detection rate and minimizing the errors. Many of such characteristics are listed by [45] [46]. A summary of the characteristics is provided in [47] and are listed below.

- Run continuously without human supervision
- If fault tolerant and be able to recover from crashes
- Is simply tailored to a specific network
- Adapts to behaviour to a specific network
- Works in real time
- Detect the maximum number of intrusions with a minimum number of false positive alarms
- Is self-monitored
- Is self-configurable to the security policies changes
- Operated while maintaining minimum overhead

With the advancements in the technologies that involve sharing of data and resources, the IDS must handle with the amount of data and the number of components that are involved in the network. With the emerging technologies in ICS such as CPS and Industry 4.0, a tremendous amount of data transfer happens. The developed IDS must compel with many such novel requirements.

3.2.1. Classification of intrusion detection system

The IDS can be classified into various categories based on their detection methods (signature/ anomaly), data source (network based/ host based), analysis timing (real-time/ offline), system architecture (centralized/ distributed), reaction after detection (passive/ active) [48] [38]. [49]also classified the IDS into categories based on the type of intruders (external/ internal) and types of intrusion (leakage/ malicious use / etc.) [50]. Three major properties of classification and their sub-categories are given in the following Fig. 5.

In this section, the IDS categories based on the detection methods and the data sources are explained in detail, as they are keenly focused in this scope of research.

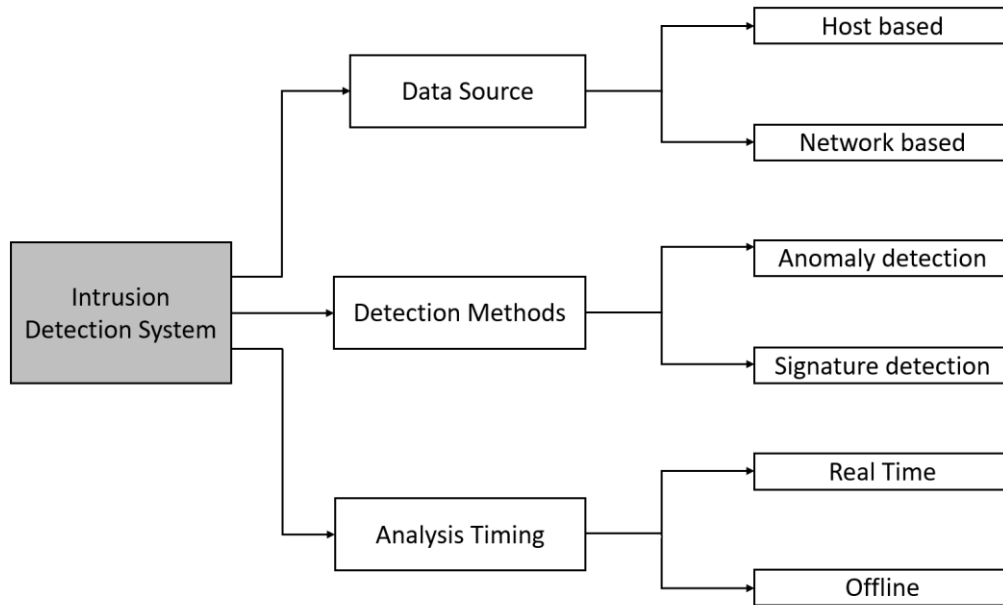


Figure 5: Classification of intrusion detection systems

From Fig. 5, we can see that IDS based on its data sources into two main categories: network based intrusion detection (NIDS) and host-based intrusion detection system (HIDS). They are simply defined as that “HIDS provides protection for the host on which it is being installed where as NIDS suspects for attacks or irregular behaviour by inspecting the contents and header information of all the packets moving across the networks” [51] [52]. A detailed discussion is given below:

NIDS – Network based Intrusion Detection System

NIDS are placed at an intentional point or points with in the network to monitor the traffic going in and out of all devices in the network [53]. NIDS are mostly passive devices that monitor the on-going network activity without adding significant overhead or interfering with the network operations. They are easy to secure against attack and may even be undetectable to attackers; they also require little effort to install and use on existing networks. Ideally it would scan all inbound and outbound traffic; however, doing so might create a bottleneck that would impair the overall speed of the network. The functional implementation of NIDS in a network infrastructure is represented in Fig. 6.

From Fig. 6, we can see that NIDS lies before the network infrastructure and all the network packets are passed via a switch into the NIDS. There every packet is analysed individually based on the implemented detection methods. If any malicious activity is found in the network, it is reported to the network administrator to take necessary action.

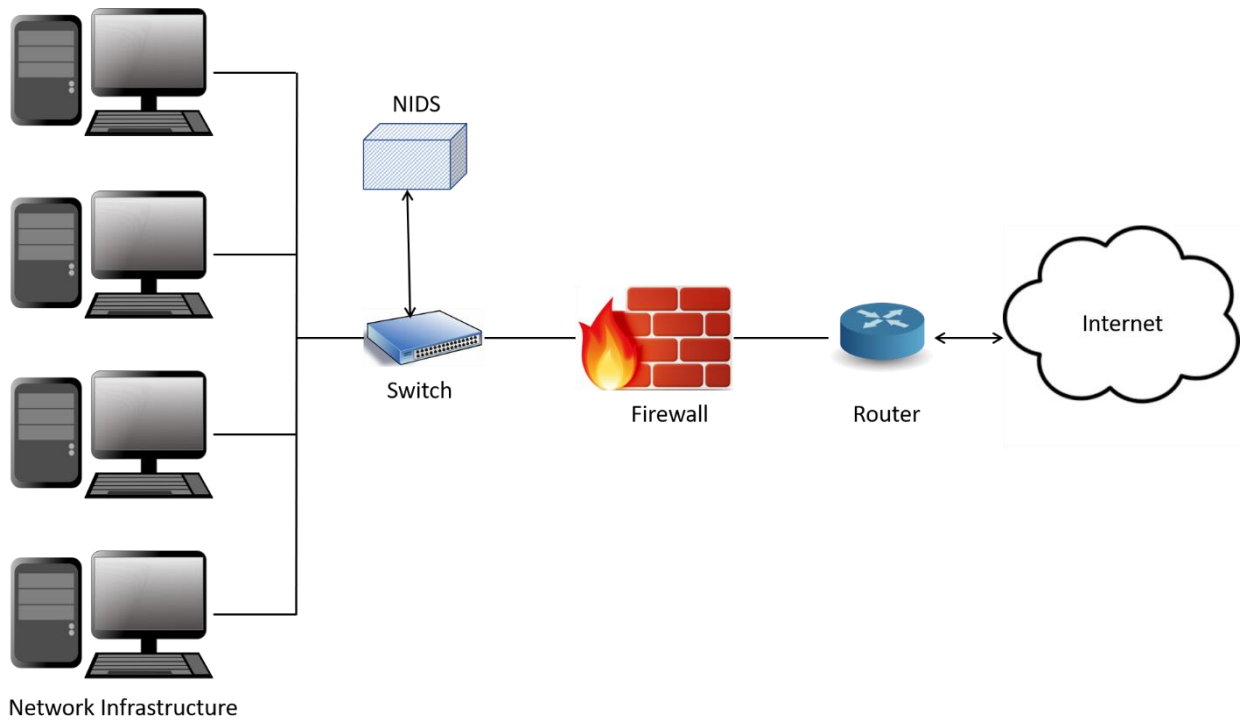


Figure 6: Functional placement of NIDS in a network infrastructure

HIDS – Host based Intrusion Detection System

HIDS runs on individual hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the user or administrator of suspicious activity detected [53]. The suspicious activities are based on the type detection technique employed. For example, audit analyse technique is able to identify the activities related to operating-system-level intrusion and application-level intrusions. The functional implementation of HIDS in a network infrastructure is represented in Fig. 7.

From Fig. 7, we can see that HIDS is like a piece of software or a hardware that lies on each and individual device on a network and all packets that are receiving to this device are analysed. If any malicious packet the user of the device or/and the administrator is informed.

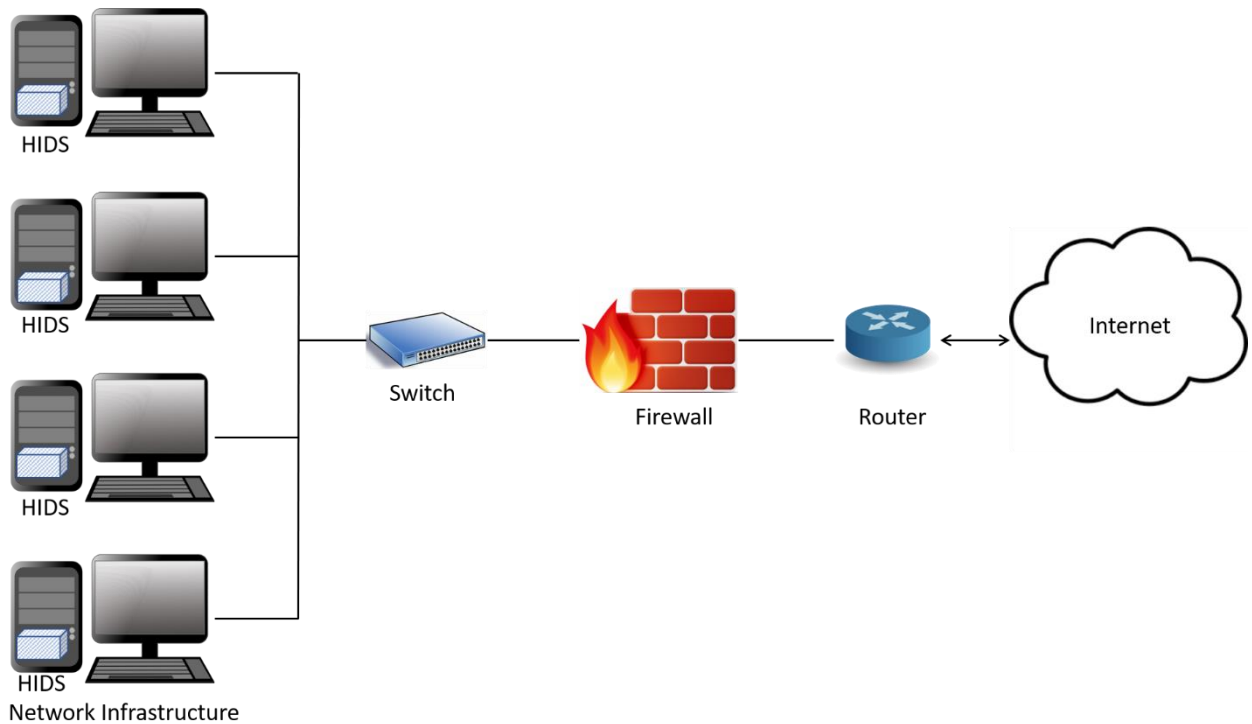


Figure 7: Functional placement of HIDS in a network infrastructure

From Fig. 5, we can see that IDS is classified based on the detection methods into two popular categories termed as signature based and anomaly detection based. In signature based intrusion detection analyse the network packets for known patterns of intrusion where as in anomaly based intrusion try to identify the novel patterns that are different from the known behaviour [54] [55]. A detailed discussion is given below:

Signature based Intrusion Detection System

A signature based IDS also termed as misuse based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats [56]. The issue is that there will be a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to your IDS. During that lag time your IDS would be unable to detect the new threat. The functionality of the signature based IDS is shown in Fig. 8.

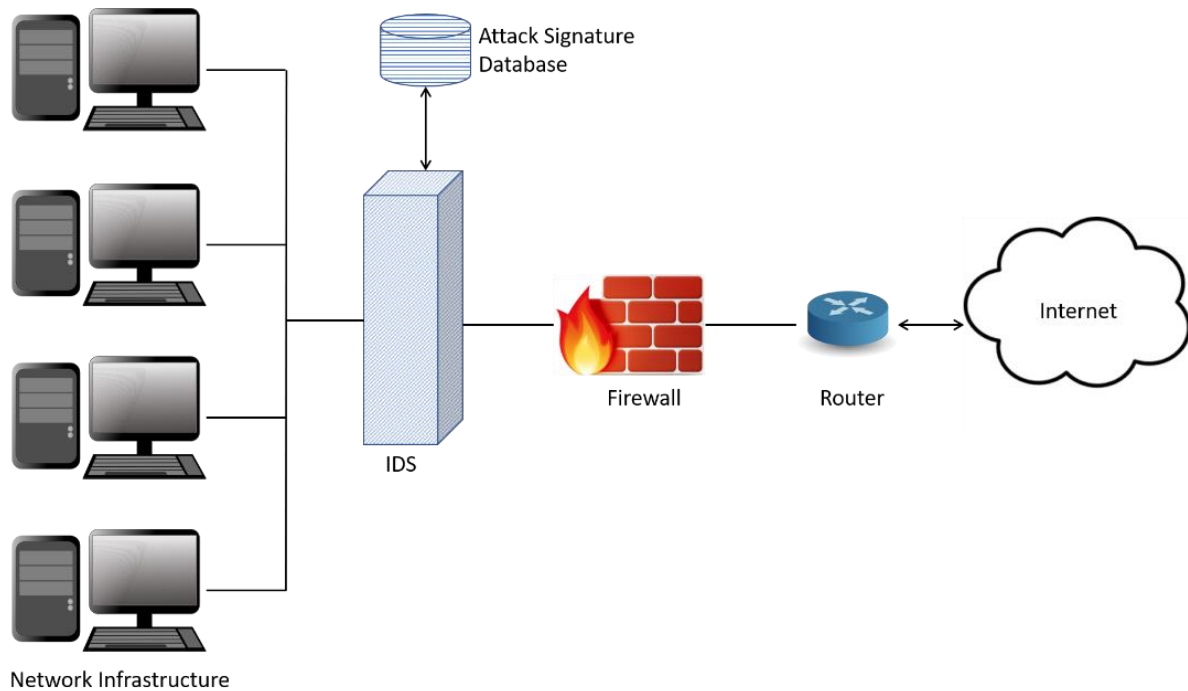


Figure 8: Functional diagram of signature based intrusion detection system

The signature based IDS works similar to an antivirus's software/hardware. That means the signature based IDS works like the virus scanners where the known identity or signature is searched for each and every intrusion attempt. It is very efficient to have signature based IDS but at the same time signature log of a network should be updated in regular basis as like antiviruses. Due to the new techniques of the attackers the signature database must be updated with the regular signatures. Therefore, the signature based IDS is good enough only if the signature database is extended on regular basis. There are two major shortcomings of signature IDS and they are:

- The signature based IDSs could be fooled very easily by changing the signature architecture of an attack. This technique normally goes around the signature database which creates an opportunity for attackers to intrude.
- The CPU load increases gradually as the signature database volume increases because it is required to check a symptom with every entry of the database. This might cause of packet dropping beyond the bandwidth capacity.

Anomaly based Intrusion Detection System

In anomaly based intrusion detection, it is assumed that the nature of the intrusion is unknown, but the intrusion will result in a behaviour that is different from the normal behaviour that is

seen in the system. This abnormal behaviour of the network is considered as anomaly. [57] defined anomalies as "the patterns in data that do not conform to a well-defined notion of normal behaviour".

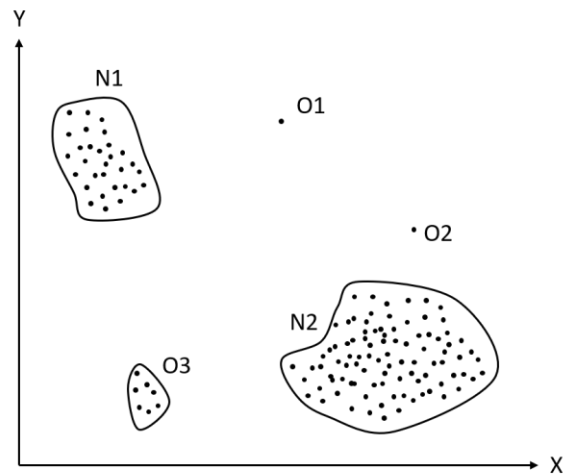


Figure 9: Illustration of anomalies in a 2-dimensiona dataset

Fig. 9. shows two normal regions $N1$ and $N2$ and most of the observed data points lie in those two regions. The data points that are far from those regions, namely point $O1$ and $O2$ and points in region $O3$ are considered as outliers termed as anomalies.

An IDS which is anomaly based will monitor network traffic and compare it against an established baseline. The baseline will identify what is "normal" for that network what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other- and alert the administrator or user when traffic is detected which is anomalous, or significantly different than the baseline [58].

In network communication the header of each and every transmitted data packet is captured by the anomaly based IDSs. These captured packet headers are filtered according to the known legal traffic. The known legal traffics are learned by the network administrator. Anomaly based IDS plays a role like security guard who takes interview of every incoming packets before the entry. This high level of filtering reduces the data quantity to be analyzed but still a big number of log data can be created by anomaly based IDSs. These log data are analyzed according to their functionality or pattern. Since almost every packet is monitored in anomaly based IDSs, it is really difficult for the attackers to send any kind of packets including malicious code. There are few drawbacks still available in anomaly based IDSs and they are described below:

- One of the major disadvantages of anomaly is to define the rules for detecting intrusion because the protocol analysis for every feature must be determined, accomplished and tested for achieving better detection accuracy. Therefore, it is a difficult job to define the rule.
- It requires more hardware resources throughout the network for anomaly based IDSs compared to the signature based IDSs.

The key differences between signature based and anomaly based intrusion detection along with their advantages and disadvantages is discussed in Table 3-1.

Some other minor categories of IDS are based on data processing techniques and through data labels.

Data processing techniques

After collecting the data, the IDS use that data to train its classifiers in order to distinguish between normal and abnormal behaviours. The IDS algorithms purpose is to report an attack. The output of the detection mechanism can be categorized into two main types [57].

- **Scores:** This is mainly used for anomaly detection techniques where an anomaly score is assigned to each tested instance that determines whether it is an anomaly or not based on a pre-defined threshold.
- **Labels:** The detection techniques assign a label (e.g. normal or abnormal or attack class) for a tested instance. Determining whether the label is normal or abnormal depends on the type of algorithm technique type.

Table 3-1: Comparison of key differences between signature based and anomaly based intrusion detection

	Signature Based Detection	Anomaly based detection
Advantages	<ul style="list-style-type: none"> • Simplest and effective method to detect attacks • Efficient in finding known attacks 	<ul style="list-style-type: none"> • Efficient in detecting new attacks • Less dependent on OS • Easy configuration
Disadvantages	<ul style="list-style-type: none"> • Difficulties in keeping signatures up to date. • Time consuming to maintain the knowledge • Long string-matching times with increase in size of the database • Not effective in detecting new attacks 	<ul style="list-style-type: none"> • Weak profiles accuracy due to observed events • Unavailable during rebuilding of behaviour profiles • Difficult to trigger alerts in right time • Not suitable for the system with regular behavioural changes

Data Labels

Every data sample is associated with a label that determines the sample either as normal or abnormal. Labelling of a normal behaviour is simple that labelling an abnormal one. It is obvious that identifying an abnormal behaviour is hard as it is very dynamic in nature so, the new types of anomalies could have no labels in the training data [57]. Based on the data labels and their availability, the detection techniques operate in different modes:

- **Supervised detection**, where the detection algorithm is trained using labels for normal and abnormal behaviour samples and patterns
- **Semi-supervised detection**, where only normal samples are labelled and the abnormal are not labelled for the training purpose of the IDS algorithm

- **Unsupervised detection**, where the samples are not required to be labelled when the detection algorithm is trained.

For each mode of the detection technique mentioned above, there are different detection algorithms that fall under it. The detection algorithms are categorized into different models. The main categories that have been used by many researches can be briefly summarized into following categories:

- **Statistical based:** These are used by anomaly detection techniques that use statistical theories to assign anomalies and their scores.
- **Instance based:** This is also called as lazy-learning algorithm [59], as it delays the generalization or induction processes until the classification process ends. K- nearest neighbour is the most popular instance-based algorithm which is based on the principle that dataset instances generally exist in close proximity to these instances with similar properties [60] [61].
- **Classification based:** This uses the training data to model a classifier that assigns intrusion scores or labels to a tested instance [57].

As mentioned in Chapter 2, the possible attacks on ICS are classified into network attacks and injection attacks. Due to this reason, literature review on the development of IDS on the network attacks and injection attacks is discussed separately in the following sub-sections.

3.3. Intrusion Detection Systems – Network Attacks

As the development of IDS date backs to early 90's, many of the research methods are out dated or not relevant to the scope of this research work. In this sub-section, the relevant literature which performs the intrusion detection using prominent machine learning techniques and deep learning algorithms using KDD dataset and NSL-KDD dataset are discussed. This research helps later in evaluating the performance of the developed deep learning algorithms.

In order to identify network attacks, some relevant features from network packet are necessary. Fig. 10 illustrates the necessary parameters from a sample TCP/IP network packet that can be used as basic features by IDS. From Fig. 10, we can see that except datagram, all the network parameters of a network packet (marked in red colour) are necessary to generate the basic feature set of a network packet.

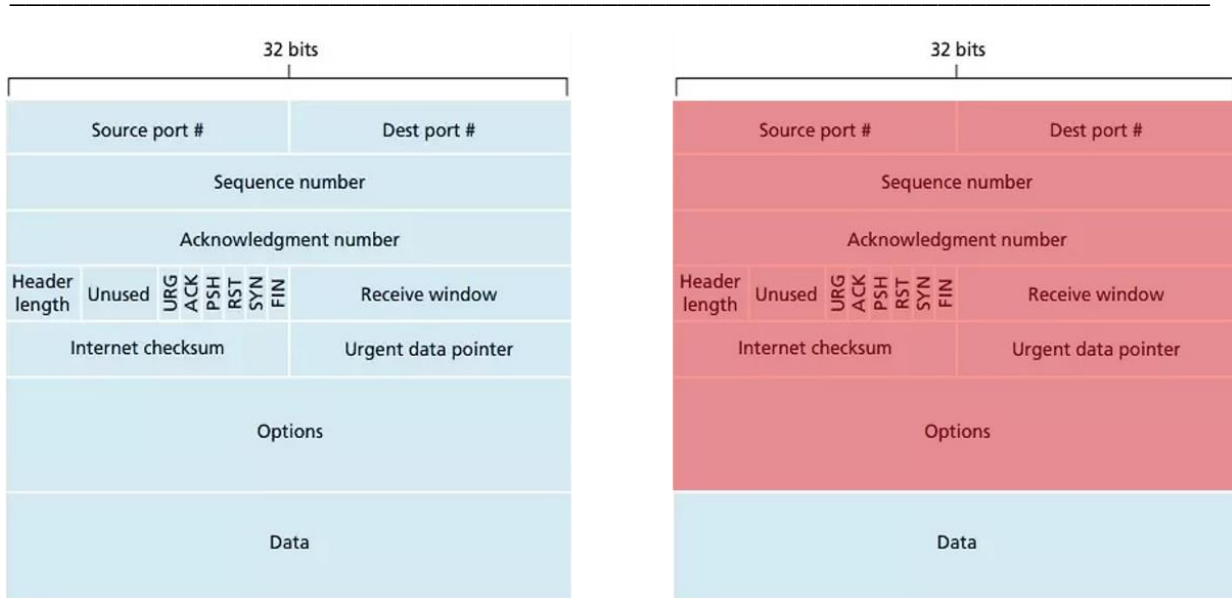


Figure 10: Architecture of network packet and parameters in network packet used for identification of network attacks

The further literature review on how these network features used by different techniques for intrusion detection are discussed below.

3.3.1. Machine learning for NIDS

[62] from Columbia University, NY, proposed that “the intrusion detection can be thought of as a classification problem: we wish to classify each audit record into one of a discrete set of possible categories, normal or a particular kind of intrusion”

Artificial Neural Networks

Artificial Neural Networks (ANN) are considered to be the technical equivalent of the neural network of the human brain. ANN structures imitate the inter-connections of different neurons similar to brain that creates a network of simple, but highly interconnected processing units which are able to computer output values gives a set of input values [63]. The traditional ANN architecture is represented in Fig. 11. From Fig. 11, we can see that a traditional ANN comprises of an input layer, a hidden layer and an output layer. All these neurons are interconnected and works as a processing units like in human brain. ANNs are one of the most familiar and common ML technique used for the development of IDS.

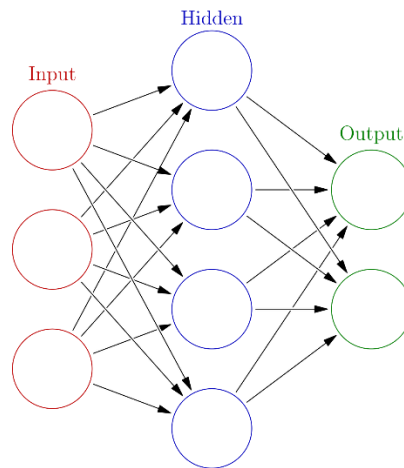


Figure 11: Traditioanl architecture of artificial neural networks

Intrusion detection literature has offered a wide spectrum of the works related to the development of IDS by adoption different structures of the ANN either for misuse or for anomaly-based detection of the malicious events. Some important research work such as [64] combines ANN with fuzzy clustering to offer a predictive model called FC-ANN, that performs adequately for low-frequent attacks. By utilizing the fuzzy clustering technique, the authors created a smaller dataset of KDDCup'99 that are given as inputs to the ANN. Using “divide and conquer” approach [65], the authors advocate that this separation of the dataset helps each ANN to achieve more precise results for the lo frequent classes of the dataset. The results of the different ANN are aggregated using a fuzzy meta-learner to deliver a complete prediction model. An intrusion detection model called MOVCCIDS is presented in [66]. This model is able to visualize the network traffic data through a functional and mobile visualization interface, which reveals the internal structure of the data. This approach can provide valuable insights to security administrators.

[67] used Self-Organizing Maps (SOM) for data clustering and Multi-Layer Perceptron (MLP) to deliver an anomaly-based IDS.

Decision Trees

Decision Trees (DT) are widely used model of supporting decision making in the context of machine learning [68]. DT are constructed as graphs, where internal nodes represent conditions for testing the attribute values of instances in a dataset with the aim of inferring a target value. This value can be either a discreet value i.e. the classes of the instance (Classification tree) or a continuous value (Regression Tree). A sample architecture of a decision tree with root nodes and child nodes is depicted in Fig. 12.

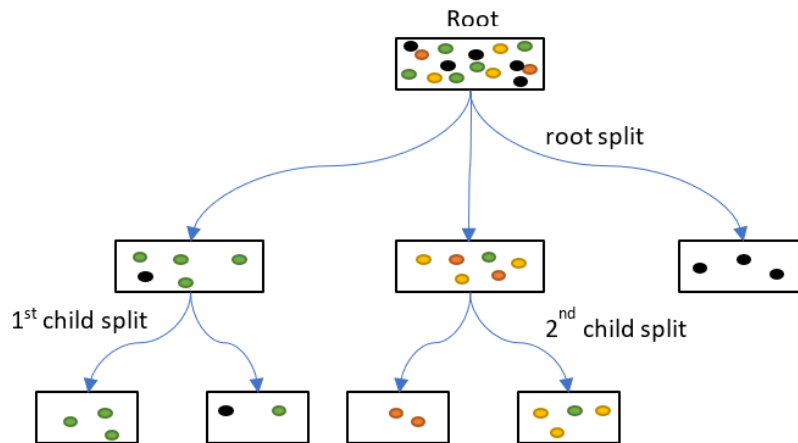


Figure 12: Architecture of decision trees

C4.5 [69], ID3 [70] and CART [71] are considered as standard decision tree algorithms that are mostly considered in the context of IDS. These algorithms are used to create a DT based on the training instance and their classification ability is measured during the testing period on previously unseen data.

DT pose some limitations and challenges such as, fail to learn minor classes and sensitive to class imbalances. DT algorithms usually generate complex structures DT and pruning techniques should be adopted to minimize the size of the trees. Several discretization techniques are also applied to the continuous data features for tree size minimization. Additionally, algorithms driven by information gain that construct a DT are biased towards major classes of a dataset achieving low classification accuracy for the classes with a smaller percentage in a dataset [72]. This drawback poses the main challenge in the context of developing IDS as the network traffic data flow has such behaviour of imbalance dataset.

DT are combined with other models in the development of IDS to improve the detection accuracy. [73] presented a lightweight IDS based on a wrapper approach and DT. The wrapper approach [74] aimed to identify an efficient subset of features in the dataset to improve the overall performance of the system. The author proposed a multiple neural network model in an assembled fashion in order to pre-process the dataset and derive a new one. The new dataset was given in C4.5 tree classifier to realize the final detection model. Additionally, the authors in [75] utilized a genetic algorithm to identify a subset of features of the KDDCup'99 dataset in order to maximize the performance of a classifier.

Support Vector Machines

Support Vector Machines (SVM) [76] are widely used machine learning based classification algorithm that can achieve high performance results based on simple concepts. The history of SVM goes back to the seventies [77]. The main objective of this ML algorithm is to find a hyperplane that distinctly classified the data points of a given dataset in an N-dimensional space.

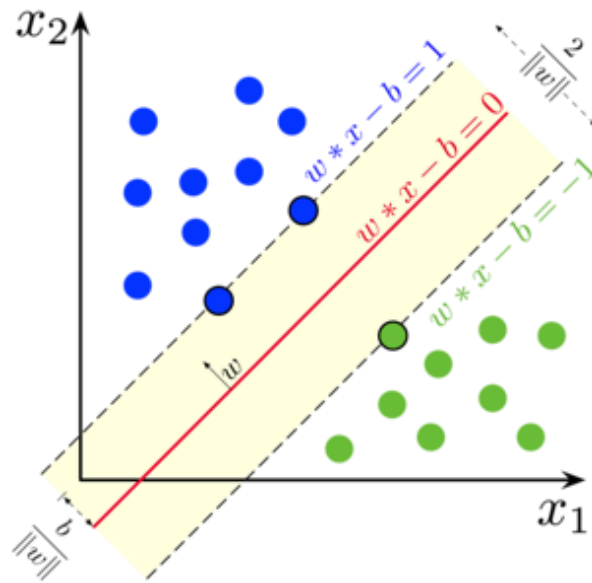


Figure 13: Sample SVM classification in a 2D dataspace

In a simple scenario, as shown in above Fig. 13, where a dataset contains two classes, the objective of an SVM is to define a plane that has the maximum perpendicular distance (i.e. margin) between the closest points of both classes. The points are called the support vectors. The wider is this margin between the hyperplane and the support vectors, the higher is the generalization ability of the predictive model.

In real data, the definition of an optimal hyperplane is a difficult task as the data may not be linearly separable. Thus, transformations are applied to augment the input dimensions of the problem with the aim of achieving the class separation. These transformations are termed as kernels. Several kernels can be used to transform the input space into higher dimensions, such as Polynomial Kernel and a Radial Kernel. This is called a Kernel Trick [78].

SVM are mainly designed to address binary classification problems. However, multi-class classification problems can be solved by breaking the problem into a multiple binary class

classification tasks and combining the final result. Common approaches in this context are the one-versus-all or pairwise classification one-versus-one approaches [79] [80].

Similar to ANN, SVM are also other common ML algorithms used for the development of anomaly-based IDS. Some important research works such as [81][82][83] uses an SVM classifier to build an anomaly IDS. The training process of the classifier is enhanced using Dynamically Growing Self Organizing Tree (DGSOT) algorithm for clustering analysis in order to pinpoint the support vectors [84]. Thus, the authors achieved faster SVM training and higher generalization that achieves higher scores in detecting attacks compared to other legacy methods. A similar approach in [85] combined BIRCH hierarchical clustering algorithms to create a smaller dataset with abstracted datapoints that aid the SVM classifier to build a more accurate model. Four SVM were trained separately, to cover all the major classes of the dataset. [86] presented an IDS which uses a genetic algorithm (GA) based optimization methodology to define an optimal feature subset and the kernel parameters (C , γ) of an SVM classifier. SVM achieves good results in the context of intrusion detection but required huge pre-processing and parameter optimization due to its complex nature and high input dimensions. In addition, the design of multi-class classification development requires the combination of multiple binary classification tasks.

There exist several other ML algorithms which are used for the development of IDS. [87] implemented a using the dimensionality reduction technique Principal Component Analysis (PCA) which is also known as Karhunen-Loève transform [88] to identify a subset of rules to classify the behaviour of the network packet as normal or abnormal. Each rule of intrusion detection is an if-then clause form and the condition part of the rule is composed of the features connected by the AND function. The result of each rule is a verification of an intrusion taxonomy. This approach has the ability to pre-process network data in real-time and offer a high detection rate and low false positive rate. Nevertheless, they considered only three kinds of attacks which are not sufficient to assess the technique.

[89] proposed the use of Bayesian Networks (BN) and Classification and Regression Trees (CART) algorithm for IDS. They used these two paradigms as a hybrid classifier and as an ensemble classifier. Initially feature selection was performed on the DARPA dataset to speed up the computation. First, BN and CART were evaluated separately with full and with a subset of features. The performance of the set of 41 features was compared to a set of 17 selected by BN and 12 selected by CART. BN performed worse with a smaller set of features except on the

normal class. CART using a reduced dataset achieved 100% normal class classification and an increased U2R and R2L classification accuracy as well. The hybrid model was able to classify the normal, probe and DOS instances more accurately.

[90] proposed a heterogeneous ensemble by fusing three different classifiers designed for detecting a single class. They adopted learning algorithms such as the Linear Genetic Programming (LGP), Random Forest (RF) and Adaptive Neuro Fuzzy Inference System (ANFIS) to construct a network based IDS. Each of the algorithms (i.e. LGP, ANFIS and RF) used the same dataset during the training. Rough Set Technique and Discrete Particle Swarm Optimisation (RST-BPSO) was employed to extract the important features. The original 41 features were cut down to 155 for all classes and selected features for each class are varied. After building the base classifiers, the system then used the weighted voting method to determine the final classification. Overall, the performance of LGP is better in comparison with the other two algorithms, while both ANFIS and RF are nearly at the same level.

Different other ML based techniques for the development of IDS can be further reviewed in the following surveys [91] [92] [93].

3.3.2. Deep Learning for NIDS

"Deep learning techniques also come under the subcategory of machine learning algorithms. But discussing about every machine learning algorithm used for the development of IDS is not possible. A detailed analysis of NSL-KDD data using various machine learning techniques with the Waikato Environment for Knowledge Analysis (WEKA) tool is discussed in [94]. Different deep learning techniques for IDS is discussed here" [95].

"Deep learning-based studies show that it completely surpasses the traditional methods in intrusion detection. In [96], deep neural networks for flow based anomaly detection was proposed and proves that deep learning techniques can be used for anomaly detection in software defined networks. [97] uses deep learning with self-taught learning technique and benchmarks the performance using NSL-KDD dataset for network intrusion detection. Here deep learning is used to classify the normal and attack classes. Performance evaluation for multiclass classification was not performed" [95].

"In [98], Recurrent Neural Networks (RNN) are considered as reduced-size networks. They classify the multiple attack classes and the performance looks promising. But the dataset used for training is not complete NSL-KDD dataset, they used a part of the training dataset, which

makes the performance biased. They also concentrated mainly on feature grouping rather than attack classification. Unfolded RNN were used in [99], and also used the limited training dataset of NSL-KDD dataset for training against attacks. When compared to existing machine learning approaches, the detection accuracies are higher with RNN. DBN for IDS was proposed by [100] and explained the efficiency of achieving higher accuracy. They performed the training operation with 20%. 30% and 40% of the NSL-KDD train dataset and tested it with the same" [95].

"Overcoming the above mentioned drawbacks, [101] uses SAE for deep feature extraction and multiclass attack classification. The results look promising and much better than the existing approaches. To overcome the drawback of long training time [102] mentioned the use of accelerated computing platform techniques to train the deep neural networks faster along with multi-class attack classification. In [103], the use of hybrid deep learning techniques a combination of deep learning and machine learning techniques were discussed. For better classification, a combination of multiple detection mechanisms with a ranking approach for highest detection accuracy of the individual attack classes was proposed" [95].

"Recently [104], provided a detailed multiclass class classification of NSL-KDD datasets using the DBN and SAE. They outperformed the detection accuracy when compared to other approaches by proposing the nonsymmetric deep autoencoder. They also performed a more detailed 13-class multi-class classification to evaluate the performance of their proposed approach and looks promising. Despite the results looks promising, they used the same training dataset to test and evaluate the performance of the proposed approach which leads to achieve higher detection accuracies" [95].

"As CNN are mainly performed on images, only one related work using CNN for development of IDS was found. We also used this approach as a basis for our implementation. [105] provided an effective image conversion method of NSL-KDD data set. The numerical features in NSL-KDD are normalized using min-max normalization and then different binary values are assigned to the different features of NSL-KDD data. This assigned binary values are converted to an image for training and testing of the CNN. This approach converts all the NSL-KDD features into image format. Even though the existing CNN approaches performed a structured pre-processing, the performance of IDS was analyzed using pre-available CNN architectures such as ResNet50 and GoogLeNet which are famous for real image processing applications. The accuracies were not satisfactory and discussion on multi-class classification fails which

also led us to investigate further on the performance on CNN for multi-class classification. Another research on CNN based IDS was mentioned in [106] used 10% KDDcup 99 dataset. Despite getting better accuracies they just used 10% dataset so this research is not considered in our benchmarking" [95].

3.4. Intrusion detection System - FDIA

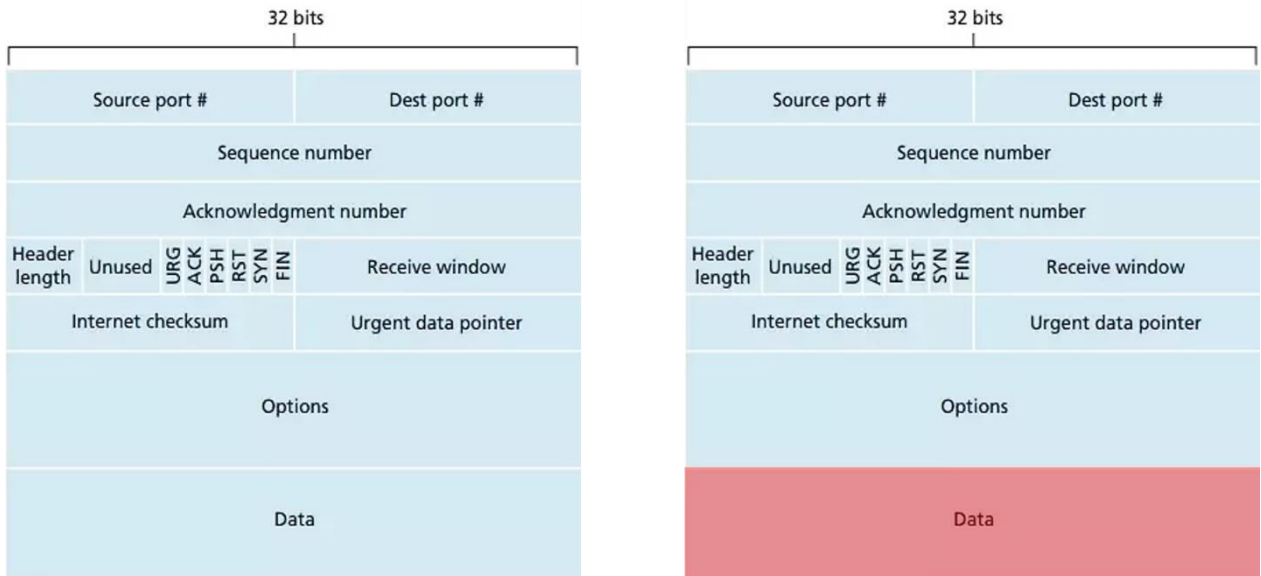


Figure 14: Archtiecture of network packet and parameters in network packet used for identification of injection attacks

In order to identify injection attacks, the datagram needs to be extracted from the network packet. Fig. 14 illustrates the structure of a network packet from which the data needs to be extracted. From Fig. 14, we can see that the data datagram (marked in red colour) is necessary to analyse for the injection attacks. From Fig. 10 & Fig. 14 we can clearly see the key difference in the analysis part for the identification of network attacks to the identification of injection attacks.

[107] introduced the mathematical model of false data injection. Assuming the real measuring vector z is represented as

$$z = \{z_1, z_2, \dots, z_m\} \tag{3-1}$$

and the observed measuring vector z_a is represented as

$$z_a = \{z_{a1}, z_{a2}, \dots, z_{an}\} \tag{3-2}$$

and if

$$a \{a_1, a_2, \dots, a_m\} \quad (3-3)$$

represents the injected false data vector, z_a will be equal to

$$z_a = z + a \quad (3-4)$$

when there is false data injection, a will not be zero vector.

In [108], the conditions for undetectable FDIA's are developed and the minimum number of sensors to be attached to ensure the undetectability is computed. [109] a linear deception attack scheme that can fool the popular X2 detector is provided.

The further literature review discusses different techniques proposed for identification of injection attacks is discussed below. As the research related to FDIA on ICS mainly concentrates on the smart grids and power sector, some important relevant literature is discussed.

"Several failure detection algorithms in dynamic systems were reviewed by the author in [110]. A Complete Survey on existing attacks and detection methods for false data injection are given in [111]. Different types of FDIA attacks such as maximum magnitude-based attack, wave based attack, positive or negative deviation attack and mixed attacks were discussed in [112]. The impact of FDIA in control systems is discussed in [27]. All the mentioned research concentrates on the impact of FDIA on ICS" [113].

"Since FDIA would result in abnormal behaviour, widely researched Anomaly Detection (AD) techniques can be applied for ICS. Due to the novel attack identification capabilities, machine learning and deep learning based AD techniques were exploited in many domains [114]. AD based network intrusion detection systems are the most commonly used techniques to identify anomalous network patterns. Many techniques such as [115] [116] use machine learning and deep learning techniques to identify the novel network attack patterns using deep learning techniques" [113].

"Anomaly based detection to detect strong attacks that feature the injection of a large amount of spurious measurement data in a very short time was provided by [117] in the smart grid applications. Three types of injection attacks were discussed in [118] who uses a network level water control system to provide a closed loop defence framework to secure cyber physical systems. Single-input, single output scheme is used to verify the performance of controlled auto-regressive moving average models is discussed in [119]" [113].

"AD techniques were also used for monitoring sensors networks and abnormal event detection. [120] mentioned the importance of anomaly detection-based approach and used a knowledge database to identify the attacks. This approach becomes complex if the amount of sensor data to be watched is enormous and requires a constant update of knowledge base which is practically not possible. A Bayesian network based approach for anomaly detection was performed in [121]. They combined Bayesian networks with Kalman Filter for predicting sensor failures but they did not consider any possible attacks on the network. Several machine learning algorithms were also used to identify the network attacks. Supervised machine learning techniques such as feed forward neural networks [122] and unsupervised learning techniques such as self-organizing maps [123] were used in identifying the attacks in network traffic. The detection accuracies of the unsupervised learning techniques were not comparable with the supervised learning mechanism when the labels are available. A further analysis is necessary to understand the outcome of the unsupervised technique which needs thorough knowledge on the process" [113].

"A lot of study on FDIA on smart grids is available [124] [125] [126]. Some other techniques such as state estimation [127], machine learning [128], sparse optimization [129] were discussed also used for identification of attacks. Our research has identified that the impact of FDIA on ICS needs to be addressed and the research on identification FDIA in industrial infrastructure is limited. A neural network based FDIA identification approach on automation plant was discussed in [130]" [113].

Efficient attack detection and secure estimation schemes for linear Gaussian systems under cyber-attack on a static, unknown sensor subset have been developed in [131], but the proposed detector is not designed to tackle the linear deception attack of [109]. The optimal attack models to steer the control of CPS to target a value is provided in [132], while ensuring a constraint on the attack detection probability, centralized and decentralized attack detection schemes for noiseless systems have been developed in [133]. [134] proposed a coding of sensor output approach to efficient attack detection using X2 detector. Attack resilient state estimation of a dynamical system with only bounded noise has been discussed in [135]. Sparsity models to characterize the switching location attack in a noiseless linear system and state recovery constraints for various attack modes have been described in [136]. Attack detection, secure estimation and control in the presence of FDI attack for power systems are addressed in [137] [138].

In recent years, intensive research activities on investigating the FDIA's which affect the integrity of a system have emerged [139] [140]. To defend against such attacks, residue based X2 detector is widely deployed [141] [139] by checking the statistic characteristics of the obtained sensor data. However, statistical approaches are not suitable in all cases and the loopholes still exist for the adversaries to perform the attacks. [107] [142] reveals that a certain category of stealthy attacks (FDIA) which can deteriorate system performance without being noticed by the proposed detectors.

3.5. Evaluation metrics of IDS

Detection rate or true positive rate and false alarm is also known as false positive rate or recall are considered to be the best metric that can be used to evaluate the developed IDS. The detection rate is equivalent to the efficiency, and the false alarm rate is equivalent to the effectiveness of the IDS. Most of the performance metrics are based on the evaluation of the on the classifiers. The evaluation of the classifier is based on the confusion matrix generated. A sample confusion matrix for the 2-class classifier is shown in Fig. 15. Fig.15 gives information about the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) through a confusion matrix. A confusion matrix is generated based on the outcomes of the trained ML and deep learning algorithms.

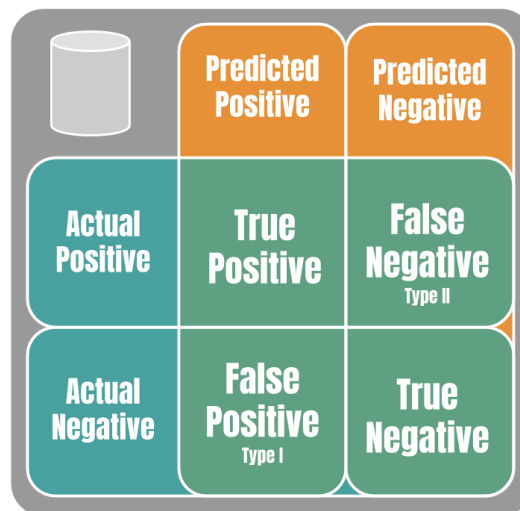


Figure 15: Confusion Matrix to understand TP, TN, FP, FN

The performance metrics that are used to evaluate all the proposed methods and calculated using the following procedure:

- i. **Accuracy:** "The number of detected samples correctly over the total number of samples, which is represented in percentages" [24].

- ii. **Precision (P):** "The precision is mentioned for Positive Predictive Value (PPV) and it is calculated in percentage via dividing the TP by the summation of TP and FP" [24] as given in equation (3-5).

$$P = \frac{TP}{TP + FP} \times 100 \quad (3-5)$$

- iii. **Recall (R):** "The recall is mentioned as the TP rate and it is calculated in percentage via TP by the summation of TP and FN" [24] as given in equation (3-6).

$$R = \frac{TP}{TP + FN} \times 100 \quad (3-6)$$

- iv. **F-Measure:** "The F-Measure is a measurement for representing the test accuracy and mentioned as harmonic mean of values P and R " [24] and can be calculated by using the equation (3-7).

$$F = \frac{2 \times P \times R}{P + R} \quad (3-7)$$

3.6. Conclusions and limitations of IDS

IDS play an important role in finding possible attacks or threats and have a significant positive impact on security infrastructure. However, it is not an answer to all issues related to security, as there are some limitations. One of those limitations is its inability to trace and analyse all traffic on highly loaded or busy networks. Therefore, the system may not be able to provide an instantaneous report for attacks or threats in such scenarios. It is also reported that IDS do not help if there is weakness in a network protocol, or in the absence of strong identification and authentication mechanism. Another limitation would be lacking the capability of conducting an investigation in the absence of human interaction. They also mentioned that it is not effective in dealing with a switched network. A study reported a number of issues with IDS. One of those issues is that some IDS do not provide verification for the checksum field in the IP header. This shortcoming gives hackers a chance to manipulate this field. As a result, the system will record different information than what it should receive. Moreover, it was found that IDSs are not cheap solutions as it consumes different types of resources during both setup and monitoring phases. In addition, it demands a high level of technical and organizational expertise. In spite of the requirements of a lot of resources and expertise, it is not simple to trace the improvement

in security processes. A common complaint reported is that IDS can generate an enormous number of alerts while the majority of those alerts are false positive.

4. Dataset

Especially addressing complex problems like anomaly-based intrusion detection using data driven approaches such as deep learning requires a dataset of a certain quality. The quality of a dataset depends on the following issues:

- How many different types of classes that are going to be classified?
- How different the classes are?
- How good are the data samples for individual classes?
- How big is the dataset to the proposed application requirements?
- Is the dataset balanced or unbalanced?

A dataset needs to provide answers to those questions. Only then it is possible to use the dataset for train and test an IDS. This chapter provides detailed information about the dataset used and generated for the process of development of IDS using deep learning algorithms.

4.1 Network attack dataset – NSL-KDD

NSL KDD dataset is one such dataset that is used to train and test an intrusion detection system. It is derived from its predecessor, KDD'99 dataset [143]. More details of the KDD'99 dataset can be seen in [144]. A web server is visited by many clients, and this results in a tremendous amount of traffic data. The traffic data is logged and can be used to analyze the behavior to be normal or abnormal traffic. For this, each network connection is mapped to a certain set of attributes, which is then analyzed using some machine learning algorithm.

KDD'99 data set was having a lot of drawbacks, which had affected the detection accuracy of many IDS. Hence NSL-KDD dataset is a refined version of the KDD'99 dataset. Compared to the KDD'99 dataset, NSL KDD dataset has following features [145],

1. The redundant records are removed to enable the classifiers to produce an un-biased result.
2. Sufficient number of records is available in the train and test datasets, which is reasonably rational and enables to execute experiments on the complete set.
3. The number of selected records from each difficult level group is inversely proportional to the percentage of records in the original KDD dataset.

In each record of the dataset, there are 41 attributes representing different features related to network security. The corresponding label assigned to each record is presented as the 42nd

attribute in the dataset. The label can be either an attack type or a normal behavior. The labels of attack type are classified into 4 attack classes namely DoS, Probe, R2L and U2R. Hence there are total 5 classes in NSL-KDD dataset. The description for each attack class is discussed in Chapter 2, Section 2.1 in accordance with [145].

More examples for each attack class can be learned from Table 4-1.

Table 4-1: Attack types groupd into their respective attack classes [145]

Attack Class	Attack Type
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_Password, Ftp_write, Imap, Phf, Mutlihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snpmguess, Snpmpgetattack, Httpunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

There are different downloadable files that are available for any research works. The details are listed as below and can be found at [146]

1. KDDTrain+.ARFF: The full NSL-KDD train set with binary labels in ARFF format.
2. KDDTrain+.TXT: The full NSL-KDD train set having attack-type labels and difficulty level in CSV format.
3. KDDTrain+_20Percent.ARFF: A 20% subset of the KDDTrain+.ARFF file.
4. KDDTrain+_20Percent.TXT: A 20% subset of the KDDTrain+.TXT file.
5. KDDTest+.ARFF: The full NSL-KDD test set with binary labels in ARFF format.

6. KDDTest+.TXT: The full NSL-KDD test set having attack-type labels and difficulty level in CSV format.
7. KDDTest-21.ARFF: It is a subset of the KDDTest+.ARFF file, which has no records with difficulty level of 21 out of 21.
8. KDDTest-21.TXT: It is a subset of the KDDTest+.TXT file which has no records with difficulty level of 21 out of 21.

As mentioned before, there are 41 attributes representing different features related to network security. Table 4-2, 4-3, 4-4 and 4-5 describes each attribute in detail.

Table 4-2: Basic features present in each network connection vector [145]

Attribute No.	Attribute Name	Description	Sample Data
1	Duration	Length of time duration of the connection	0
2	Protocol_type	Protocol used in the connection	Tcp
3	Service	Destination network service used	ftp_data
4	Flag	Status of the connection – Normal or Error	SF
5	Src_bytes	Number of data bytes transferred from source to destination in single connection	491

Dataset

6	Dst_bytes	Number of data bytes transferred from destination to source in single connection	0
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 , else 0	0
8	Wrong_fragment	Total number of wrong fragments in this connection	0
9	Urgent	Number of urgent packets in this connection.Urgent packets are those packets with the urgent bit activated	0

Table 4-3 : Content related features present in each network connection vectore [145]

Attribute No.	Attribute Name	Description	Sample Data
10	Hot	Number of 'hot' indicators in the content w hich could be:entering a system directory, creating programs and executing programs	0
11	Num_failed_logins	Count of failed login attempts	0
12	Logged_in	Login Status: 1 if successfully logged in; otherwise 0	0

Dataset

13	Num_compromised	Number of 'compromised' conditions	0
14	Root_shell	1 if root shell is obtained; otherwise 0	0
15	Su_attempted	1 if 'su root' command attempted or used; otherwise 0	0
16	Num_root	Number of 'root' accesses or number of operations performed as a root in the connection	0
17	Num_file_creations	Number of file creation operations in the connection	0
18	Num_shells	Number of shell prompts	0
19	Num_access_files	Number of operations on access control files	0
20	Num_outbound_cmds	Number of outbound commands in an ftp session	0
21	Is_hot_login	1 if the login belongs to the 'hot' list i.e., root or admin; else 0	0
22	Is_guest_login	1 if the login is a 'guest' login; otherwise 0	0

Dataset

Table 4-4: Time related traffic features present in each network connection vector [145]

Attribute No.	Attribute Name	Description	Sample Data
23	Count	Number of connections to the same destination host as the current connection in past two seconds	2
24	Srv_count	Number of connections to the same service (port number) as the current connection in past two seconds	2
25	Serror_rate	The percentage of connections that have activated the flag (attribute 4) s0, s1, s2 or s3, among the connections aggregated in count (attribute 23)	0
26	Srv_serror_rate	The percentage of connections that have activated the flag (attribute 4) s0, s1, s2 or s3, among the connections aggregated in Srv_count (attribute 24)	0
27	Rerror_rate	The percentage of connections that have activated the flag (attribute 4) REJ, among the connections aggregated in count (attribute 23)	0

Dataset

28	Srv_rerror_rate	The percentage of connections that have activated the flag (attribute 4) REJ, among the connections aggregated in Srv_count (attribute 24)	0
29	Same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (attribute 23)	1
30	Diff_srv_rate	The percentage of connections that were to different service, among the connections aggregated in count (attribute 23)	0
31	Srv_diff_host_rate	The percentage of connections that were to the different destination machines among the connections aggregated in srv_count (attribute 24)	0

Table 4-5: Host based traffic features present in a network connection vector [145]

Attribute No.	Attribute Name	Description	SampleData
32	Dst_host_count	Number of connections having the same destination host IP address	150
33	Dst_host_srv_count	Number of connections having the same port number	25
34	Dst_host_same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in Dst_host_count (attribute 32)	0.17
35	Dst_host_diff_srv_rate	The percentage of connections that were to different service, among the connections aggregated in Dst_host_count (attribute 32)	0.03
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same service, among the connections aggregated in Dst_host_srv_count (attribute 33)	0.17
37	Dst_host_srv_diff_host_rate	The percentage of connections that were to the different destination machines, among the connections aggregated in Dst_host_srv_count (attribute 33)	0

Dataset

38	Dst_host_serror_rate	The percentage of connections that have activated the flag (attribute 4) s0, s1, s2 or s3, among the connections aggregated in Dst_host_count (attribute 32)	0
39	Dst_host_srv_serror_rate	The percentage of connections that have activated the flag (attribute 4) s0, s1, s2 or s3, among the connections aggregated in Dst_host_srv_count (attribute 33)	0
40	Dst_host_rerror_rate	The percentage of connections that have activated the flag (attribute 4) REJ, among the connections aggregated in Dst_host_count (attribute 32)	0.05
41	Dst_host_srv_rerro_rate	The percentage of connections that have activated the flag (attribute 4) REJ, among the connections aggregated in Dst_host_srv_count (attribute 33)	0

The different attributes discussed in the Table IV, V, VI and VII can be grouped together based on the attribute value type. The 3 different attribute types present in NSL-KDD dataset are,

1. Nominal
2. Binary
3. Numeric

The classification of different attributes can be further learned from Table 4-6.

Table 4-6: Attributes and their value types [145]

Type	Feature
Nominal	Protocol_type (2), Service (3), Flag (4)
Binary	Land (7), logged_in (12), root_shell (14), su_attempted (15), is_host_login (21), is_guest_login (22)
Numeric	Duration (1), src_bytes (5), dst_bytes (6), wrong_fragment (8), urgent (9), hot (10), num_failed_logins (11), num_compromised (13), num_root (16), num_file_creations (17), num_shells (18), num_access_files (19), num_outbound_cmds (20), count (23), srv_count (24), serror_rate (25), srv_serror_rate (26), rerror_rate (27), srv_rerror_rate (28), same_srv_rate (29), diff_srv_rate (30), srv_diff_host_rate (31), dst_host_count (32), dst_host_srv_count (33), dst_host_same_srv_rate (34), dst_host_diff_srv_rate (35), dst_host_same_src_port_rate (36), dst_host_srv_diff_host_rate (37), dst_host_serror_rate (38), dst_host_srv_serror_rate (39), dst_host_rerror_rate (40), dst_host_srv_rerror_rate (41)

Table 4-7 will give the details for normal and attack data in different types of NSL-KDD dataset.

Table 4-7: Normal and attack data in different types of NSL-KDD dataset [145]

Dataset Type	Total No. of					
	Records	Normal	DoS	Probe	R2L	U2R
KDD Train+ 20%	25192	13449	9234	2289	209	11
		53.39 %	36.65 %	9.09 %	0.83 %	0.04 %
KDD Train+	125973	67343	45927	11656	995	52
		53.46 %	36.46 %	9.25 %	0.79 %	0.04 %
KDDTest+	22544	9711	7458	2421	2754	200
		43.08 %	33.08 %	10.74 %	12.22 %	0.89 %

Further study and analysis of KDDTrain+ dataset can be understood from Table 4-8. It has exposed one of the important facts about the attack class network vectors [5].

Table 4-8: Network protocols used by various attacks [145]

Attack class Protocol	DoS	Probe	R2L	U2R
TCP	42188	5857	995	49
UDP	892	1664	0	3
ICMP	2847	4135	0	0

From Table 4-8, it can be concluded that most of the attacks planned by the hackers use the TCP protocol suite. This is due to the fact that the TCP protocol is very transparent and easy to use. Hence this protocol is easily exploited by the attackers to launch network-based attacks on victim machines.

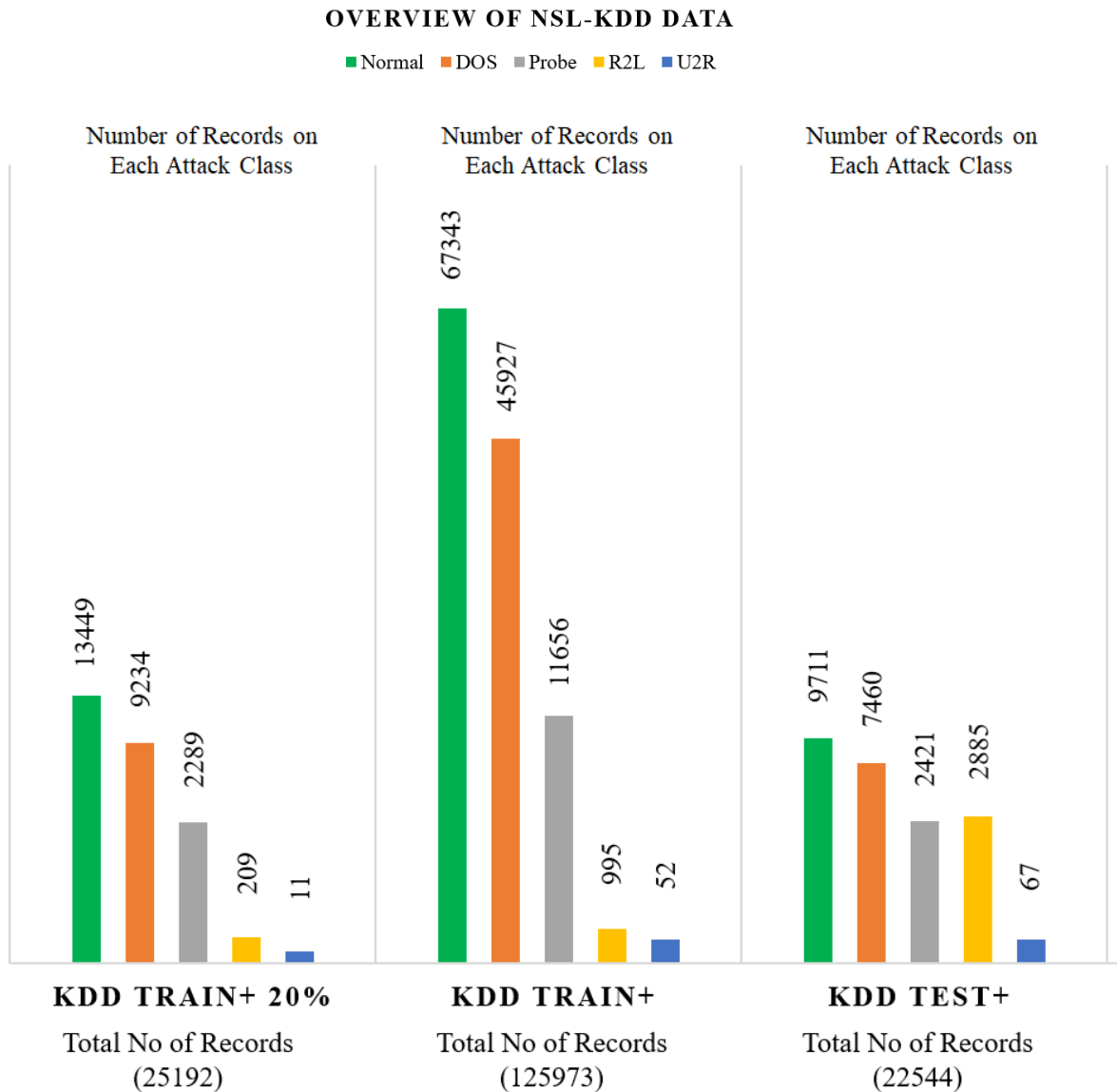


Figure 16: Visualization of no. of records in NSL-KDD datasets

The NSL-KDD data has four different types of attacks excluding normal as mentioned earlier. These attacks are attempted normally using different network protocols. Therefore, a statistic is made on the network protocols those are used for attempting different attacks and it is shown in Fig. 16.

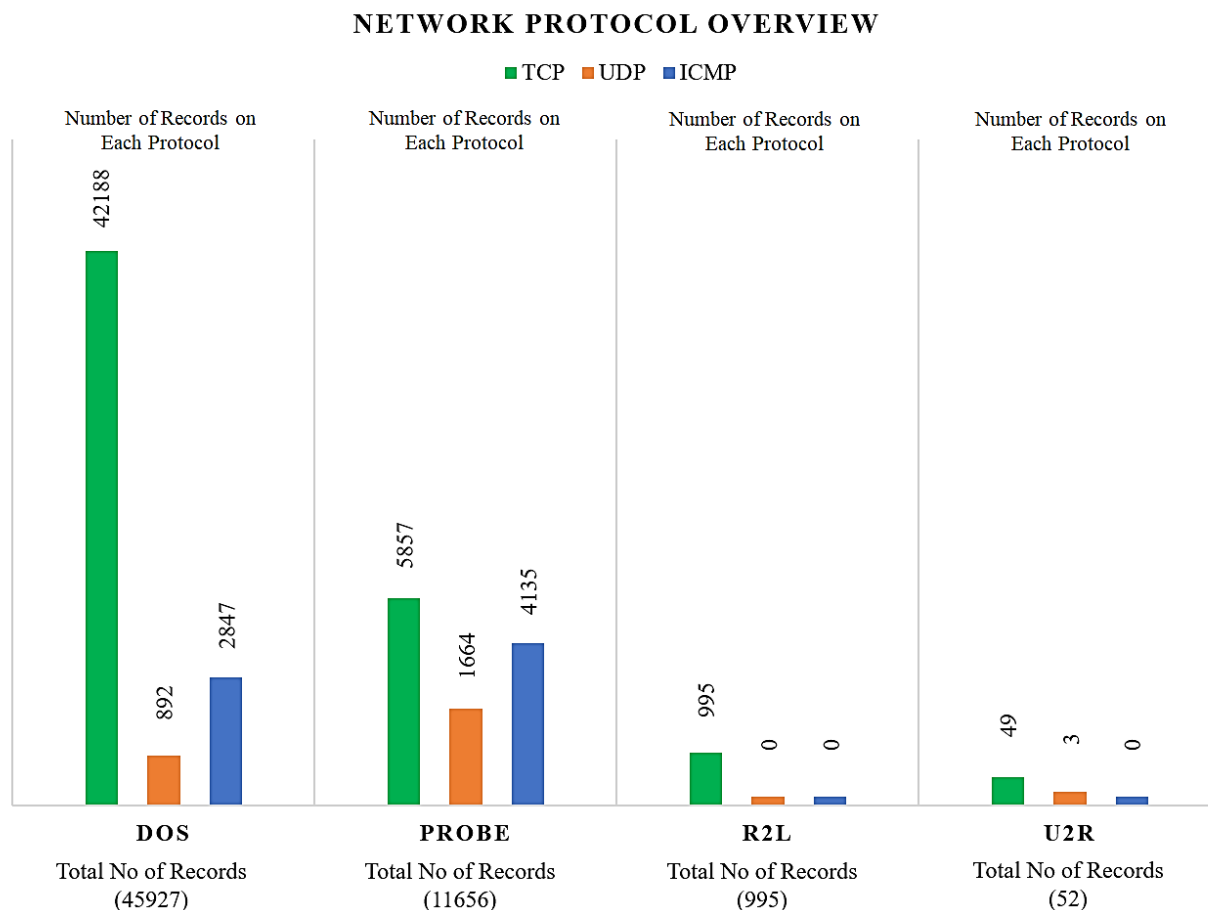


Figure 17: Used network protocols by different attack types

4.2. Network attack dataset – UNSW-NB-15

"In the year 2015 [147] [148], the UNSW-NB-15 data set was introduced first time by Moustafa and Slay. The data set was created in the lab of the Australian Centre for Cyber Security (ACCS) and the IXIA PerfectStorm tool was used to create the data set. A volume of 100 GB raw network traffic was captured using tcpdump tool to create the data set" [95]. The data set is downloadable at [149]. The UNSW-NB 15 dataset is included 49 features in total and it has nine attack categories excluding the normal. The attack classes along with their sub-classes of UNSW-NB 15 data are given in Table 4-9 on the next page and the nine attack categories are described below:

1. **Analysis:** A various type of intrusion method that stabbed the web applications using web scripts, emails and ports.
2. **Backdoors:** This is a malware type attack where; the normal authentication process of a system is paralyzed to gain access to that system.

3. **DoS:** Denial of Service makes a network system extremely busy keep sending data packets to prevent the legal authentication requests.
4. **Exploits:** It is a type of which is attempted by taking advantages of the weakness of a system. It could be approached through software libraries or application plugins.
5. **Fuzzers:** It is a black-box testing method for identifying the unknown vulnerabilities of a system.
6. **Generic:** An attack type that uses hash functions to establish a collision against the block cipher of a system.
7. **Reconnaissance:** It is similar to a probe attack and it collects information about a network or system to escape its monitoring control cleverly.
8. **Shellcode:** Shellcode is used to exploit a network vulnerability having limited access to that network. It is a small piece of code applied as payload to penetrate.
9. **Worms:** The worms are also as like malwares that reproduces itself to spread out through the other network devices connected in the same network taking advantage of security failures.

The main objective of creating this new dataset is to overcome the shortcomings of previously defined datasets like KDD Cup 99, NSL-KDD or DARPA 98/99. The efficiency of an IDS is entirely dependent on how well it has been trained to capture intrusions and the training efficiency is depended on the dataset that contains contemporary activities of normal and attack. There are three major disadvantages of previously defined datasets [148]. They are:

- "Lack of knowledge on modern footprint attack fashions. For example, the attack behaviors are changed closer to normal behavior with the time for attacks like stealthy or spy attacks" [95].
- "The defined normal traffic benchmark is not similar with the present normal traffic because these datasets were defined before two decades ago" [95].
- "The training and testing set have different distribution on attack types. For instance, the existing benchmark datasets have different data types comparing in between the training and testing set" [95].

Table 4-9: Attack sub-classes of UNSW-NB 15 data

Attack Class	Attack Type	Total Number of Types
Analysis	HTML, Port Scanner, Spam	3
Backdoors	-	0
DOS	Asterisk, Browser, CUPS, Cisco Skinny, Common Unix Print System (CUPS), DCERPC, DNS, Ethernet, FTP, HTTP, Hypervisor, ICMP, IGMP, IIS Web Server, IMAP, IRC, ISAKMP, LDAP, Microsoft Office, Miscellaneous, NTP, NetBIOS/SMB, Oracle, RDP, RTSP, SIP, SMTP, SNMP, SSL, SunRPC, TCP, TFTP, Telnet, VNC, Windows Explorer, XINETD	36
Exploits	All, Apache, Backup Appliance, Browser, Browser FTP, Cisco IOS, Clientside, Clientside Microsoft, Clientside Microsoft Media Player, Clientside Microsoft Office, Clientside Microsoft Paint, DCERPC, DNS, Dameware, Evasions, FTP, ICMP, IDS, IGMP, IMAP, Interbase, LDAP, LPD, MSSQL, Microsoft IIS, Miscellaneous, Miscellaneous Batch, NNTP, Office Document, Oracle, PHP, POP3, PPTP, RADIUS, RDesktop, RTSP, SCADA, SCCP, SIP, SMB, SMTP, SOCKS, SSH, SSL, SunRPC, TCP, TFTP, Telnet, Unix 'r' Service, VNC, WINS, Web Application, Webserver	53

Dataset

Fuzzers	BGP, DCERPC, FTP, HTTP, OSPF, PPTP, RIP, SMB, Syslog, TFTP	10
Generic	All, HTTP, IXIA, SMTP, SIP, Superflow, TFTP	7
Reconnaissance	DNS, HTTP, ICMP, MSSQL, NetBIOS, SCTP, SMTP, SNMP, SunRPC, SunRPC Portmapper (TCP) TCP Service, SunRPC Portmapper (TCP) UDP Service, SunRPC Portmapper (UDP), SunRPC Portmapper (UDP) TCP Service, SunRPC Portmapper (UDP) UDP Service, Telnet	15
Shellcode	AIX, BSD, BSDi, Decoders, FreeBSD, HP-UX, IRIX, Linux, Mac OS X, Multiple OS, NetBSD, OpenBSD, SCO Unix, Solaris, Windows	15
Worms	-	0

The UNSW-NB 15 data is available online only in “CSV” format and the downloadable files are as given below

- **UNSW-NB15_features.csv:** The available features of UNSW-NB 15 data along with their numbers and data types are described in this file.
- **UNSW-NB15_GT.csv:** The ground truth table of UNSW-NB 15 data along with attack names and references are provided.
- **UNSW-NB15_LIST_EVENTS.csv:** The list of events is provided with their numbers. Such as attack categories and their subcategories.
- **UNSW-NB15_1.csv:** The 1st portion of two million and 540,044 sample records are available.
- **UNSW-NB15_2.csv:** The 2nd portion of two million and 540,044 sample records are available.
- **UNSW-NB15_3.csv:** The 3rd portion of two million and 540,044 sample records are available.

- **UNSW-NB15_4.csv:** The 4th portion of two million and 540,044 sample records are available.
- **UNSW_NB15_training-set.csv:** A partition of UNSW-NB 15 data as training set including 175,341 samples.
- **UNSW_NB15_testing-set.csv:** A partition of UNSW-NB 15 data as testing set including 82,332 samples.

The available 49 features of UNSW-NB 15 data are further categorized into five classes except the 48th and 49th features because the 48th feature is the categorical label of the attacks where, normal is labeled as blank and the 49th feature is the binary label of attacks where, “0” represents normal and “1” represents all other types of attack. All the five categories of available features have been elaborated in following Tables 4-10 to Tables 4-14.

Table 4-10: The flow features of UNSW-NB 15 data [148]

Feature No	Feature Name	Description	Sample Data
1	srcip	IP address of the source	59.166.0.0
2	sport	Port Number of the Source	1390
3	dstip	IP address of the destination	149.171.126.6
4	dsport	Port Number of the destination	53
5	proto	Type of the protocol	udp

The details of basic features are provided in Table 4-11 below:

Dataset

Table 4-11: The Basic features of UNSW-NB 15 Data [148]

Feature No	Feature Name	Description	Sample Data
6	state	Indication of the state and its protocol dependency	CON
7	dur	Record of the duration in total	0.001055
8	sbytes	Bytes from source to destination	132
9	dbytes	Bytes from destination to source	164
10	sttl	Lifetime source to destination	31
11	dttl	Lifetime from destination to source	29
12	sloss	Retransmission or dropped source packets	0
13	dloss	Retransmission or dropped destination packets	0
14	service	Used service Protocol	ftp
15	Sload	Bits per seconds on source	500473.9
16	Dload	Bits per Second on destination	621800.9375

Dataset

17	Spkts	Number of packets from source to destination	2
18	Dpkts	Number of packets from destination to source	2

The details of Content features are provided in Table 4-12 below:

Table 4-12: The content features of UNSW-NB 15 data [148]

Feature No	Feature Name	Description	Sample Data
19	swin	The TCP window advertisement of source	0
20	dwin	The TCP window advertisement of destination	0
21	stcpb	The sequence no. of TCP based source	0
22	dcpb	The sequence no. of TCP based destination	0
23	smeansz	The mean of flow packet size (Transmitted by source)	66
24	dmeansz	The mean of flow packet size (Transmitted by destination)	82
25	trans_depth	The pipeline depth representation	0
26	res_bdy_len	Actual uncompressed content size of the data transferred from the server's http service	0

Dataset

The details of time related features are provided in Table 4-13 below:

Table 4-13: The Time features of UNSW-NB 15 Data [148]

Feature No	Feature Name	Description	Sample Data
27	Sjit	Source jitter (in mSec)	9.89101
28	Djit	Destination jitter (in mSec)	10.682733
29	Stime	Starting time of the record	1421927414
30	Ltime	Last time of the record	1421927414
31	Sintpkt	Source interpacket arrival time (in mSec)	0.017
32	Dintpkt	Destination interpacket arrival time (in mSec)	0.013
33	tcprtt	The round trip time of TCP setup	0
34	synack	The setup time for TCP connection (time between the SYN and the SYN_ACK packets)	0
35	ackdat	The setup time for TCP connection (time between the SYN_ACK and the SYN packets)	0

The details of generated other features are provided in Table 4-14 below:

Table 4-14: The Additional Generated features of UNSW-NB 15 [148]

Feature No	Feature Name	Description	Sample Data
36	is_sm_ips_ports	If srcip (1) equals to dstip (3) and sport (2) equals to dsport (4), this variable assigns to 1 otherwise 0	0
37	ct_state_ttl	No. for each state (6) according to specific range of values of sttl (10) and dttl (11)	0
38	ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service	0
39	is_ftp_login	If the ftp session is accessed by user and password then 1 else 0	0
40	ct_ftp_cmd	No of flows that has a command in ftp session	0
41	ct_srv_src	No. of records that contain the same service (14) and srcip (1) in 100 records according to the ltime (26)	3
42	ct_srv_dst	No. of records that contain the same service (14) and dstip (3) in 100 records according to the ltime (26)	7
43	ct_dst_ltm	No. of records of the same dstip (3) in 100 records according to the ltime (26)	1

Dataset

44	ct_src_ltm	No. of records of the srcip (1) in 100 records according to the ltime (26)	3
45	ct_src_dport_ltm	No of records of the same srcip (1) and the dsport (4) in 100 records according to the ltime (26)	1
46	ct_dst_sport_ltm	No of records of the same dstip (3) and the sport (2) in 100 records according to the ltime (26)	1
47	ct_dst_src_ltm	No of records of the same srcip (1) and the dstip (3) in in 100 records according to the ltime (26)	1

There are four types of data values are available in UNSW-NB 15 dataset namely: nominal, binary, numeric and time stamps. The features and their data types with the samples are given in Table 4-15 along with the feature numbers inside the braces.

Table 4-15: Different data type features of UNSW-NB 15 Data

Type	Feature	Total
Nominal	scrip(1), dstip(2), proto(5), state(6), service(14), attack_cat(48)	6
Binary	is_sm_ips_ports(36), is_ftp_login(39), Label(49)	3
Numeric	sport(2), dsport(4), dur(7), sbytes(8), dbytes(9), sttl(10), dttl(11), sloss(12), dloss(13), Sload(15), Dload(16), Spkts(17), Dpkts(18), swin(19), dwin(20), stcpd(21), dtcpb(22), smeansz(23), dmeansz(24), trans_depth(25), res_bdy_len(26), Sjit(27), Djit(28), Sinpkt(31), Dinpkt(32), tcprtt(33), synack(34), ackdat(35), ct_state_ttl(37), ct_flw_http_mthd(38), ct_ftp_cmd(40), ct_srv_src(41), ct_srv_dst(42), ct_dst_ltm(43), ct_src_ltm(44), ct_src_dport_ltm(45), ct_dst_sport_ltm(46), ct_dst_src_ltm(47)	38
Time Stamp	Stime(29), Ltime(30)	2

The UNSW-NB 15 data is saved into four parts due to storing capacity and the first three parts are prepared for training set where each part is contained with 700001 data samples and the fourth part is prepared a testing set and it is included of 440044 data samples. An overview of the prepared train and test datasets have been given in the Fig. 18. where, an analysis of the sample records has been made for all the available attack classes in the dataset and it is provided differently for training and testing parts of the dataset.

OVERVIEW OF UNSW-NB 15

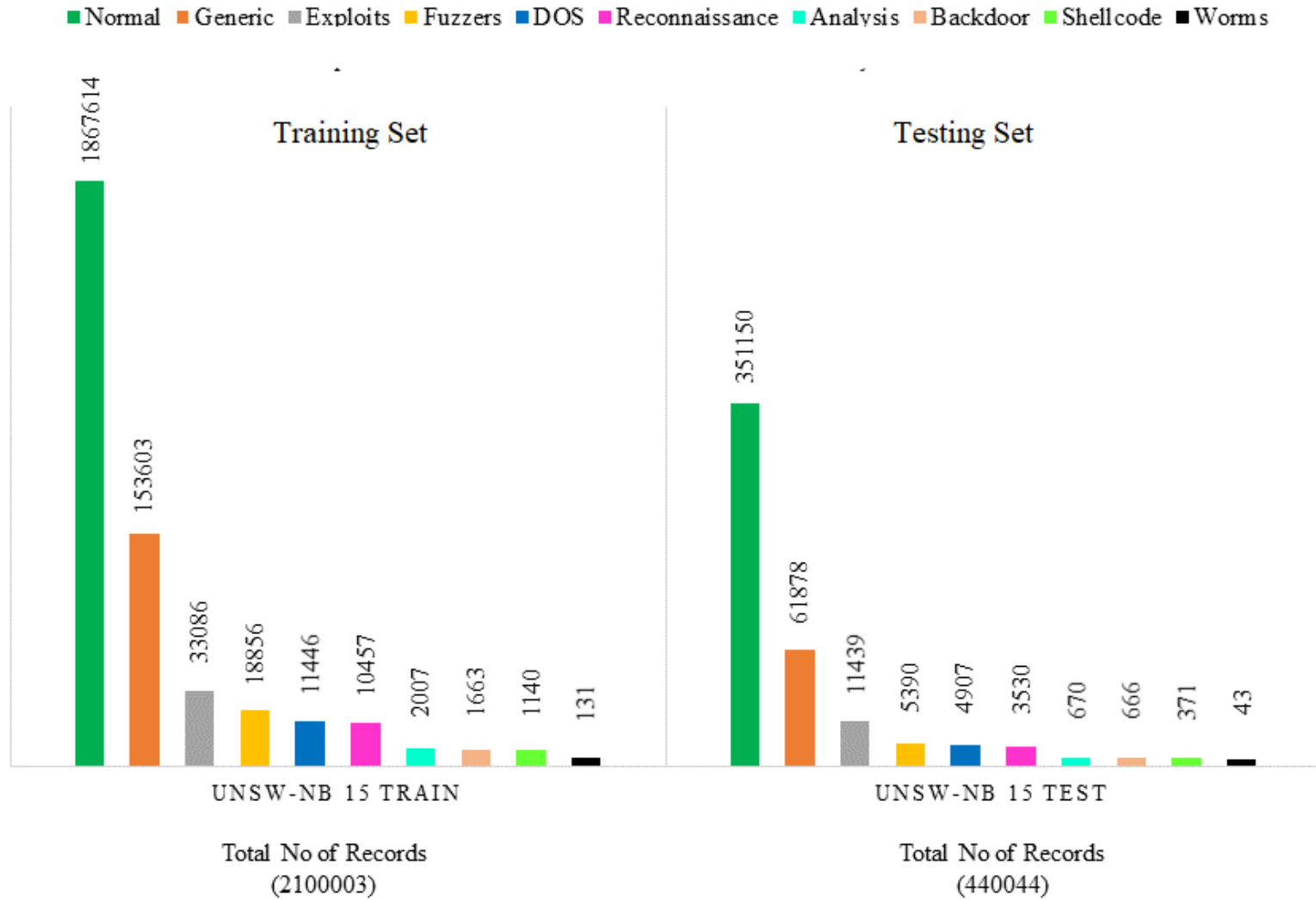


Figure 18 : The Sample Records of UNSW-NB 15 Train and Test Data

4.3. Injection attack dataset - process control plant

To analyse the functionality of the developed IDS against injection attacks, a process control use case is selected. Existing datasets such as Tennessee Eastman Process (TEP) Simulation dataset [150] [151] cannot be used in this scope of the thesis. The TEP dataset is explicitly used of the anomaly detection evaluation. The disturbance in the process variable are doesn't reflect the real attacks but represents the anomaly in dataset. The injection attacks are an intelligent way of manipulating the process variable that almost looks real and hard to distinguish from the normal variables. The anomalies in the TEP dataset also doesn't represent the injection attack behaviours on ICS mentioned in [20].

So, in order to simulate the injection attack dataset for the scope of this thesis, the chair of “Integrated Automation” at Otto-von-Guericke University Magdeburg has a Process Control Plant in the Automation Laboratory is used. This plant was built by Festo Didactic and termed as MPS PA compact workstation with level, flow rate, pressure and temperature controlled system [152]. The following Fig. 19. gives an overview about the picture of the plant and the P&ID diagram of the plant.

Using a corresponding controller such as Samson Trovisi 6495, the level and flowrate-controlled system can be set up as a cascade control system. The level of each tank is measured via ultrasound sensors and the flow rate between the tanks are measured via flow sensors. Pressure sensor and temperature sensor also provided but they are not used for this experiment. A two-way ball valve with a pneumatic process actuator which connects the elevated tank and lower reservoir is used to control the manipulated variable. The pump is controlled via speed adjustment. The level sensor values as well as the pump actuation values from the plant are used for the evaluation. Simulated measurement injection attacks were injected in the level sensor data as well as command injection attacks were injected in pump data.

The attacks were simulated very effectively that they are hard to identify in general. For example, Fig. 20. (a) represents the filtered tank level data and Fig. 20. (b) represents the attack injected signal (measurement injection). We can see some specific differences between the two signals in Fig. 20 (a) & (b) but it is hard to identify by just looking at the signal that there are some attacks injected.

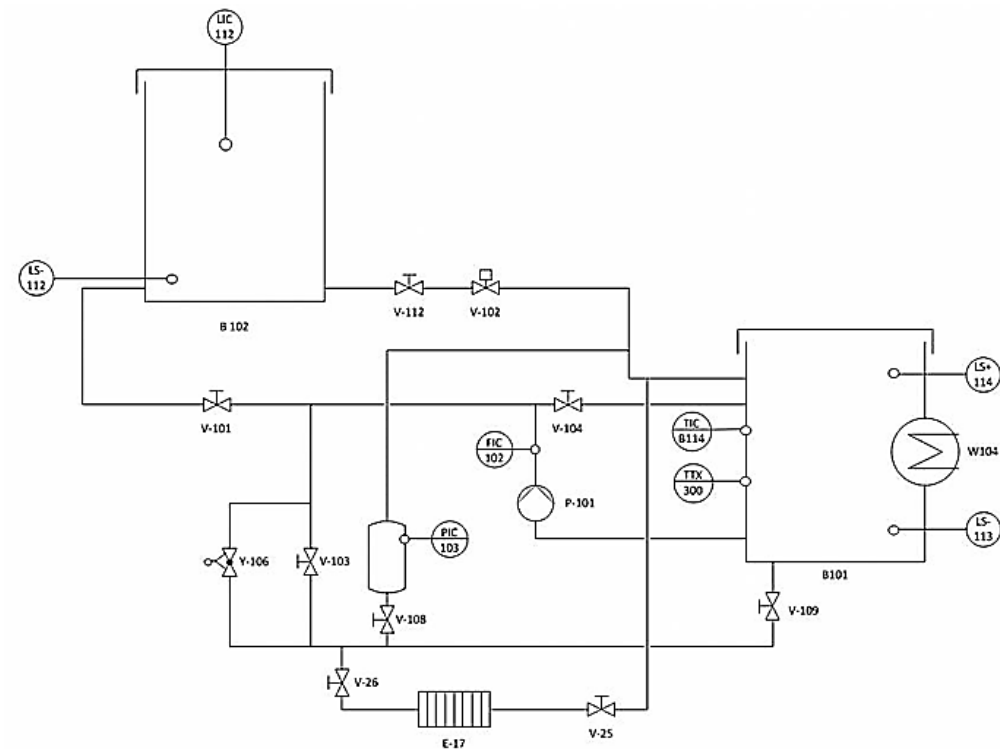
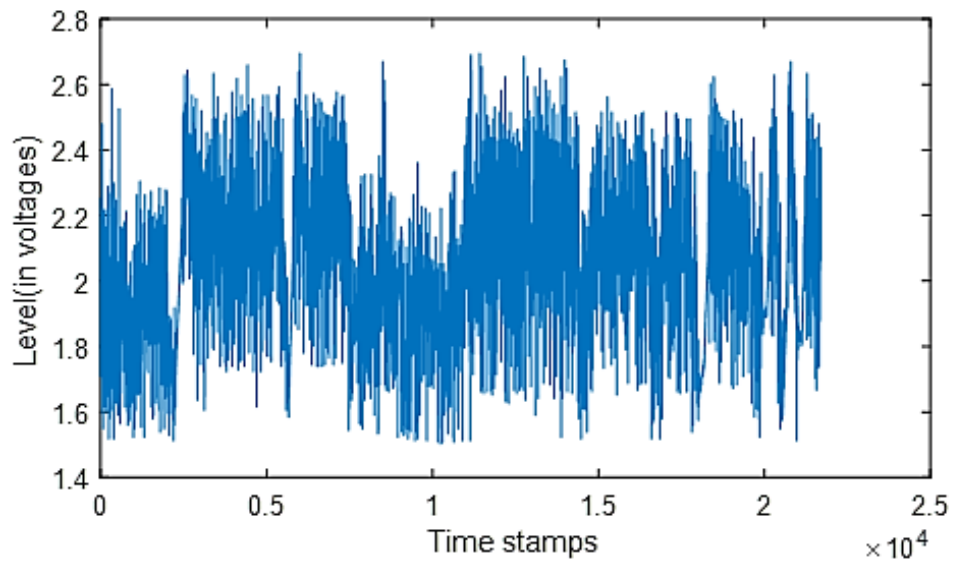
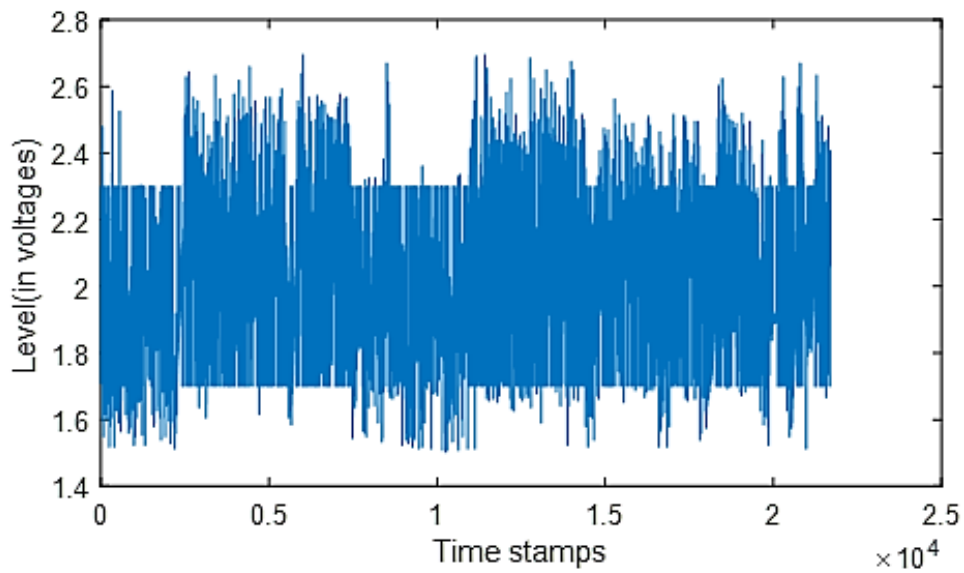


Figure 19: Process control plant and P&ID diagram of the plant

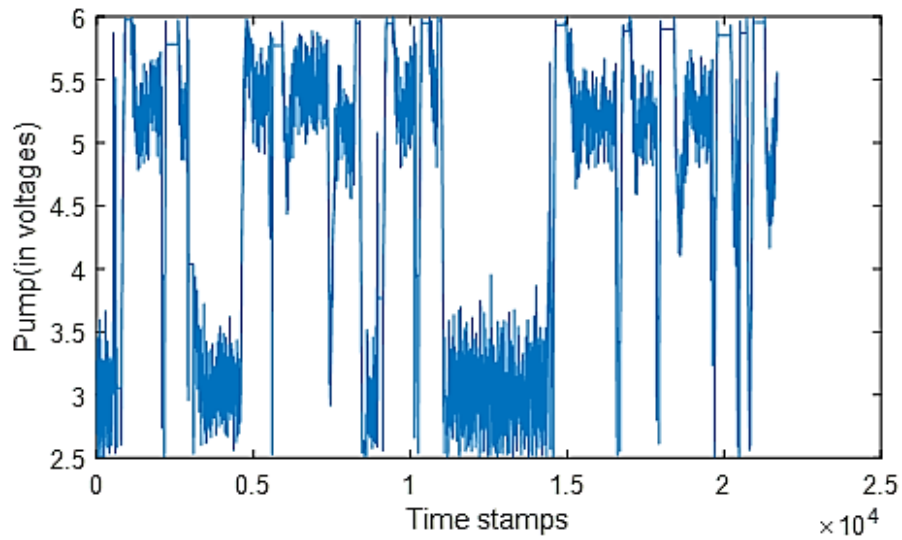


(a)

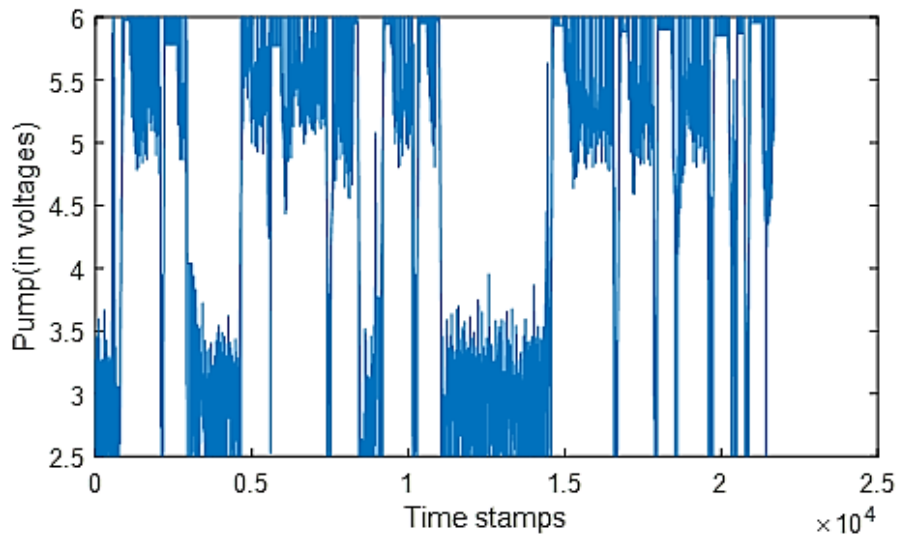


(b)

Figure 20: Tank level sensor data (a) Filtered signal (b) Attack injected signal



(a)



(b)

Figure 21: Pump data (a) Filtered signal (b) Attack injected signal

Similar to measurement injection, Fig. 21. (a) represents the filtered pump data and Fig. 21. (b) represents the attack injected signal (command injection). We can also see some specific differences between the two signals in Fig 21 (a) & (b) but it is hard to identify by just looking at the signal that there are some attacks injected. The injected attacks will be in normal range of the

signal but are inserted in a specific pattern such that the values are complete opposite to the normal behaviour but within the normal range and only through specific features the difference is identified. The level sensor data is the simulated data from the level sensors of the twin tank system shown in Fig. 19. The level sensors are used to read the liquid level in the tank constantly for the proper function of the two-tank system, but this system behaviour can be manipulated with faulty measurement injection to the sensor data by the attackers. Suppose, fault measurement data has been injected to the level sensor data then the scenario could be like this: due to fault measurement injection the sensor would read different level of liquid than the actual and caused the malfunctioning of the system. This malfunctioning could bring a great damage to the system and surrounding the system. After obtaining the sensors signals (with injections) they are used to train and test with the proposed deep learning security schema for attack identification and classification.

5. Concept and usage of deep learning algorithms

In this chapter, a detailed overview about the concept and usage of deep learning algorithms for securing ICS is presented.

5.1. Steps of data flow from ICS to attack classification

The schematic steps of the data flow from the ICS to attack classification is shown in Fig. 22. The flow defines the necessary steps in developing cybersecurity strategies to secure an ICS against attacks. This flow elaborates on different steps involved in the development process of using a deep learning for security purposes. A detailed description of the concept in the development of the deep learning algorithms in securing ICS against cyber-attacks is given in the following subsections.

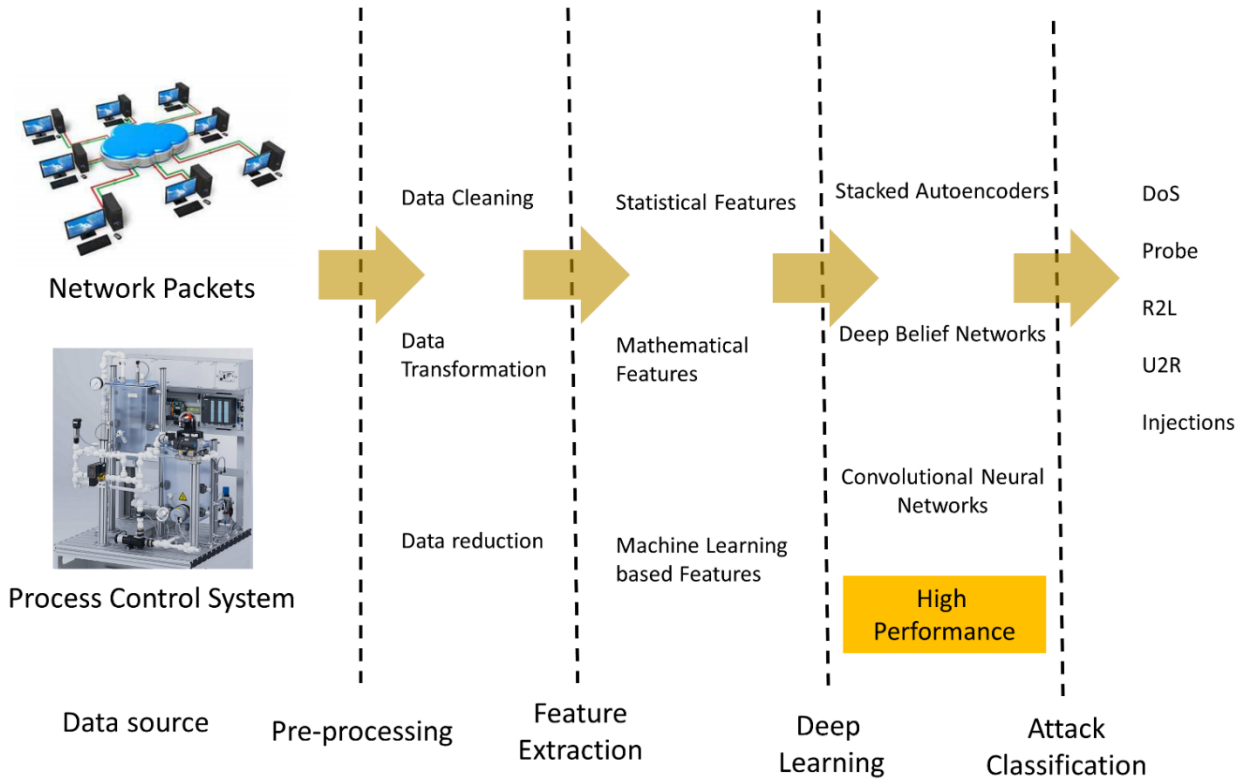


Figure 22: Schema for implementation of deep learning in securing ICS

5.2. Data source

As mentioned earlier, ICS is a complex system and encompasses several types of control systems and associated instrumentation used in the process control. In order to develop an efficient deep learning algorithm for securing ICS, two different ICS scenarios are considered. In terms of network attacks (e.g. a benchmark data set) and injection attacks (e.g. simulated process control plant data) are used for evaluation. From Fig. 22. the benchmark dataset used for evaluation of IP based networks relates to the enterprise network and service networks and the process control system gets the data from the lower level sensor, actuators or device network. A detailed description of the datasets from network infrastructure and process control application used in the scope of this thesis is discussed in Chapter 4.

5.3. Pre-processing

"Pre-processing of data is a crucial step for various applications, not only for deep learning but also for many data driven applications. The effort on pre-processing depends on the characteristics of acquired data. It involves transforming raw data into a deep learning algorithm understandable format. The data obtained from real-world is often incomplete, inconsistent, lack certain common behaviour or trend and is likely to contain errors and corrupted values. Pre-processing techniques such as data cleaning, data transformation and data reduction are proven techniques to resolve the above-mentioned issues. Data cleaning detects and corrects corrupt or inaccurate records in the collected data. This includes removing of corrupted data which may occur due to equipment malfunction, handling of noisy data and outlier removal. Data transformation includes tasks such as smoothing, normalization, aggregation and generalization of acquired data. Normalization is the key task in data transformation which scales the data to a specified range. Min-Max normalization and z-score normalization techniques are most commonly used normalization techniques. Data reduction is usually done when acquired data is too big to handle or work with. Dimensionality reduction and aggregation are two common approaches in data reduction. A more detailed information about the data pre-processing can be found in [153]. Pre-processing prepares the data for further processing, such as feature extraction" [24]. Below gives a detailed overview of the performed pre-processing procedures on the individual datasets.

5.3.1. Pre-processing of NSL-KDD dataset

"As deep learning algorithms can take only numerical values for training and testing, a pre-processing stage is necessary to convert the non-numerical attributes to numerical values. Two main tasks of pre-processing NSL-KDD dataset are" [154]:

- "Converting the non-numerical attributes in the dataset to numerical values. The features 2, 3 and 4 represent the protocol type, service and flag. These attributes in the NSL-KDD train and test data set were converted to numerical values" [154]. The conversion of non-numeric features to numeric features sample for protocol type feature is given in Table 5-1.
- "Converting the attack name at the end of the dataset into its numeric categories. 1,2,3,4 and 5 were assigned to normal, DoS, Probe, R2L and U2R respectively" [154].

Table 5-1: Representation of non-numerical characters to numeric in NSL-KDD dataset

Feature Value	Numeric Value
TCP	1
UDP	2
ICMP	3

"Since the features of the NSL-KDD dataset have either discrete or continuous values, the ranges of the feature values were different and this made them incomparable. Therefore, the features were normalized by using min-max normalization shown in Eq. 5-1 to map all the different values for individual attributes to range between [0 ,1]" [154].

$$x_{i[0 \text{ to } 1]} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (5-1)$$

Where x_i represented each data point, x_{min} is the minimum among all data points, x_{max} is the maximum among all data points and $x_{i[0 \text{ to } 1]}$ is the normalized data point value between 0 and 1.

5.3.2. Pre-processing of UNSW-ND-15 dataset

UNSW-NB 15 dataset also consists of some non-numerical data. "Hence a pre-processing stage is necessary to convert the non-numerical attributes to numerical values. Two main tasks of pre-processing are" [154]:

- "Converting the non-numerical attributes in the dataset to numerical values. The features that represent the protocol type, state and service are non-numeric. These attributes in the UNSW-NB-15 train and test data set were converted to numerical values" [154]. The conversion of non-numeric features to numeric features sample for service feature is given in Table 5-5.
- "Converting the attack name at the end of the dataset into its numeric categories. 1,2,3,4 and 5 and so on to normal, Generic, Exploits, Fuzzers and so on respectively" [154].

Table 5-2: Representation of non-numerical characters to numeric in UNSW-NB-15 dataset

Feature Value	Numeric Value	Feature Value	Numeric Value
-	1	pop3	8
dhcp	2	radius	9
dns	3	smtp	10
ftp	4	snmp	11
ftp-data	5	ssh	12
http	6	ssl	13
irc	7		

"Since the features of the UNSW-NB-15 dataset have either discrete or continuous values, the ranges of the features value were different and this made them incomparable. Therefore, the features were normalized by using min-max normalization shown in Eq. 7-1 to map all the different values for individual attributes to range between [0,1]" [154].

5.3.3. Pre-processing injection attack dataset

The sensors signal measured from the real process control plant incorporate noise in it. For analysis of data and to identify the attacks, the sensor signal should be free from the noise. As a part of pre-processing step, the noise in sensor data needs to be filtered. Filtering techniques are applied to filter those noise. The filtered sensor data is then fed for attack injection model to create the dataset with attacks presented in Chapter 4.

5.3.4. Pre-processing for implementing CNN

The implementation of a security mechanism using CNN is different from the other deep learning algorithms like SAE and DBN. The input to a CNN algorithm in general is an image or more technically as 2D or a 3D matrix of numerics. Other algorithms take mostly the data as a 1D vector or an array. Due to this reason, a separate pre-processing step is necessary to convert the dataset into a specific format to use it with CNN.

NSL-KDD dataset

The NSL-KDD data is comprised of three types of features as mentioned in according to Table 4-6 and they are nominal, binary and numeric. As the deep learning algorithms can be implemented using numeric values, a numeric or binary value are needed to assign for nominal values. The available nominal values on NSL-KDD data have been assigned with the number of bits equal to their available categories. For instance, the 2nd attribute of NSL-KDD data includes three types of nominal values namely "tcp", "udp" and "icmp" therefore, they are assigned with three bits value like tcp = 100, udp = 010 and icmp = 001. Similarly, this binary bits assignment can be done automatically using the hot encoder defined in Fig. 23 where the Y represents the number of available nominal values that will be assigned with Y number of binary bits.

Deep learning algorithms require data scaling for better generalization of a high variance data points otherwise it may cause a knock-on effect on the learning process. As explained earlier for

the use of NSL-KDD dataset with CNN, the numeric values have been normalized using min-max normalization using Equ.5-1. With this normalization, all the values are scaled in between 0 to 1.

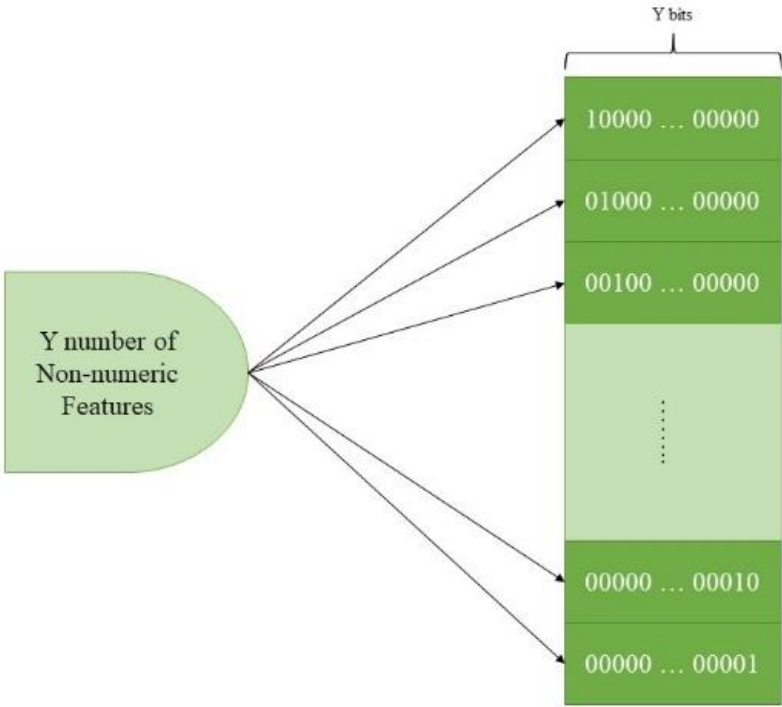


Figure 23: Hot encoder for binary extraction

All the continuous features, including the binary and normalized features, are then discretized into 10 intervals using a binary discretizer as shown in Fig. 24. The discretization is done with a standard scaler of value 0.1 and then all the ten intervals have been assigned with ten different binary bits by a hot encoder.

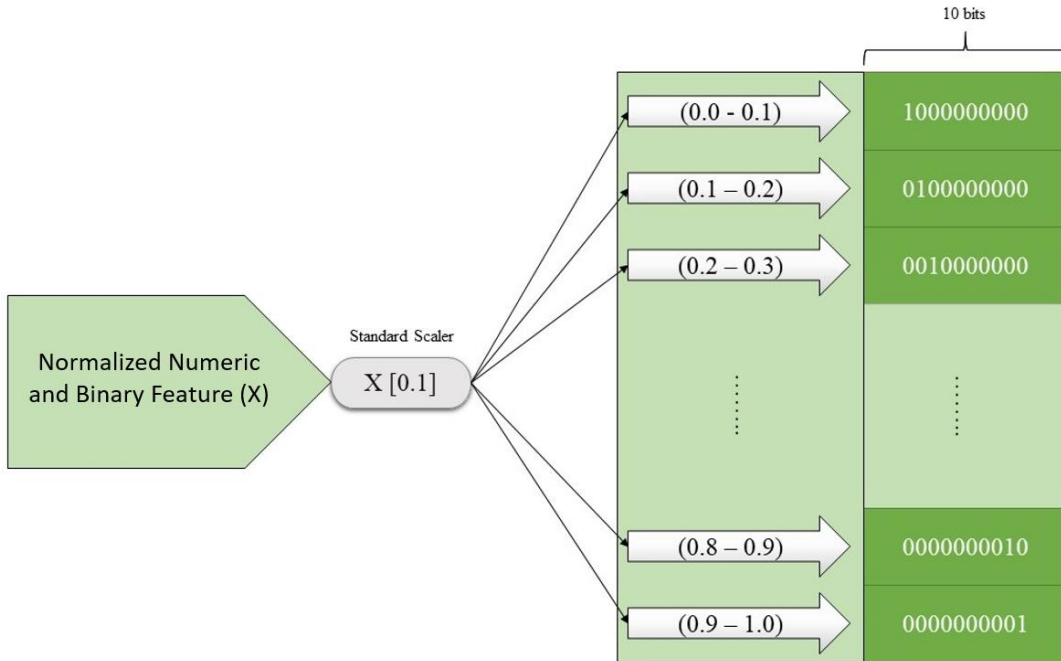


Figure 24: Discretizer for Binary Extraction of feature

The 41 features of the NSL-KDD data have been converted into a binary vector of size 464 dimensions after pre-processing where the nominal values have been assigned with 84 bits and continuous values with 380 bits.

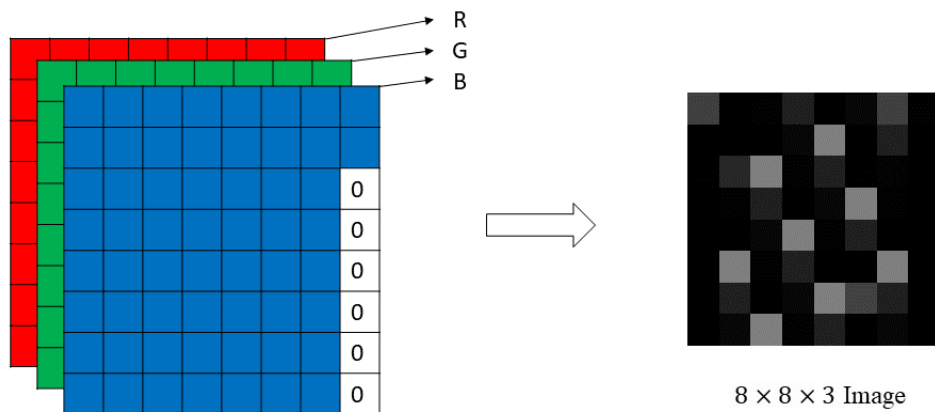


Figure 25: Image representation of NSL-KDD dataset

Each 8 bits of the binary vector of 380 bits have been converted into grayscale pixels. The converted pixels are reshaped into a 8×8 pixel matrix where the empty pixels at the last column

are padded with zero values and then this same 8×8 pixel matrix is assigned to the red green and blue channels to represent as image as shown in Fig. 25.

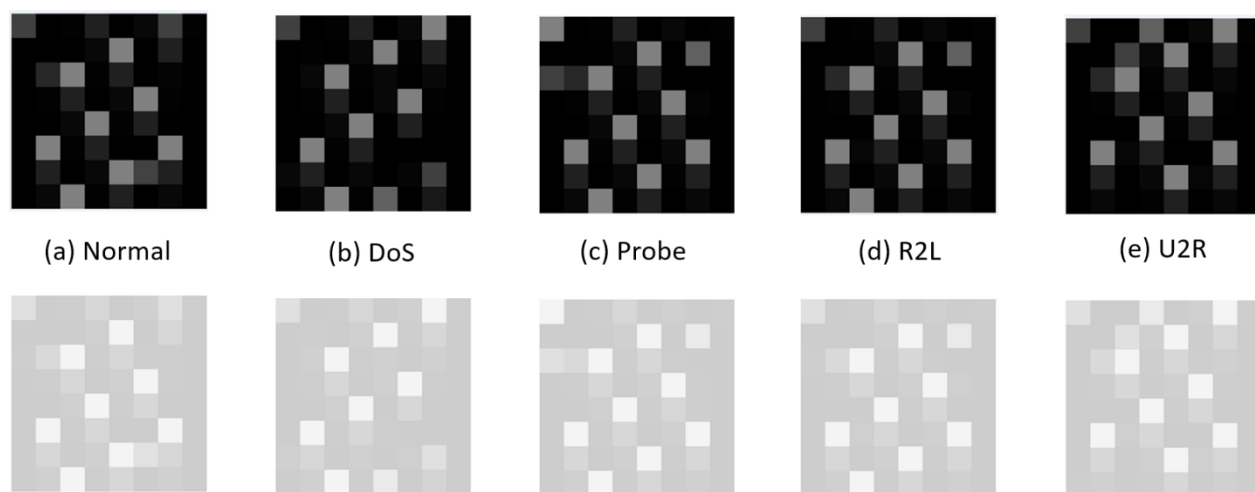


Figure 26: Input image 8x8 of different classes from NSL – KDD Dataset

The generated images for normal and different attack types in NSL-KDD dataset are represented in Fig. 26. These images are just a sample from the dataset. In order to get a deep insight into the images, a recoloured image is also presented below the original image to depict the differences between the normal and different attack types. These images are fed to CNN for training and later for testing.

UNSW-NB-15 dataset

The UNSW-NB-15 data is comprised of four data types, nominal, numeric, binary and time stamps, as mentioned in Table 4-16. The time stamps and few of the nominal features like IP addresses and port numbers are not useful; therefore, they have been removed during pre-processing. The selected nominal features of this data have been assigned with binary values using the same hot encoder that is used for NSL-KDD data in Fig. 23. The number of categories available under a feature is assigned with that number of bits. For instance, the service feature is included of 13 categories of nominal value therefore, each category of service feature has been assigned with 13 binary bits.

The numeric features of UNSW-NB-15 data have been normalized using the Eq. 5-1 for scaling purpose as it is discussed for NSL-KDD data. The normalized values are then discretized into 20 intervals by using the same discretizer used for sensor data and each interval has been assigned with 20 different bits for binary extraction.

The features of UNSW-NB-15 data are converted into a size of 924 binary bits after hot encoding and discretization. Now, for converting each 8 bits into pixels the binary vector is short of 4 bits hence, these 4 bits are filled with zeros to convert the binary bits vector into 121 grayscale pixels. These 121 grayscale pixels are arranged into 12×12 grayscale pixels matrix where, the vacant pixels are padded with zeros as like in Fig. 27. Finally, the same 12×12 pixel matrix has been sent to the red green and blue channels to represent UNSW-NB 15 data images.

The generated images for normal and different attack types in UNSW-NB-15 dataset are represented in Fig. 28 & 29. These images are just a sample from the dataset.

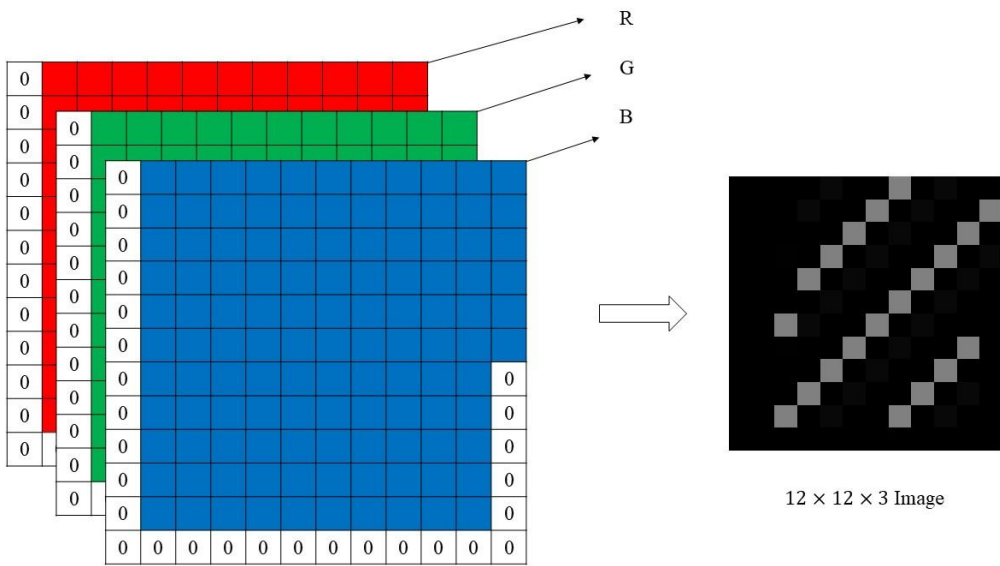


Figure 27: Image representation of UNSW-ND-15 dataset

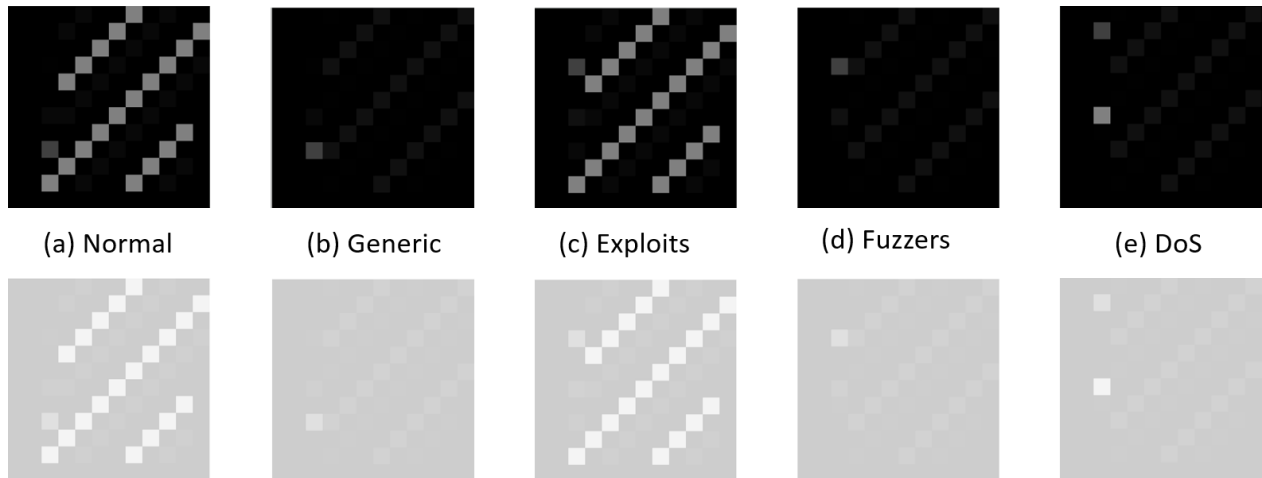


Figure 28 : Input image 8x8 of different classes from UNS-NB-15 dataset – Set 1

In order to get a deep insight into the images, a recoloured image is also presented to depict the differences between the normal and different attack types. A deep insight into the image can show a slight difference between the normal and different attack types. These images are fed to CNN for training and later for testing.

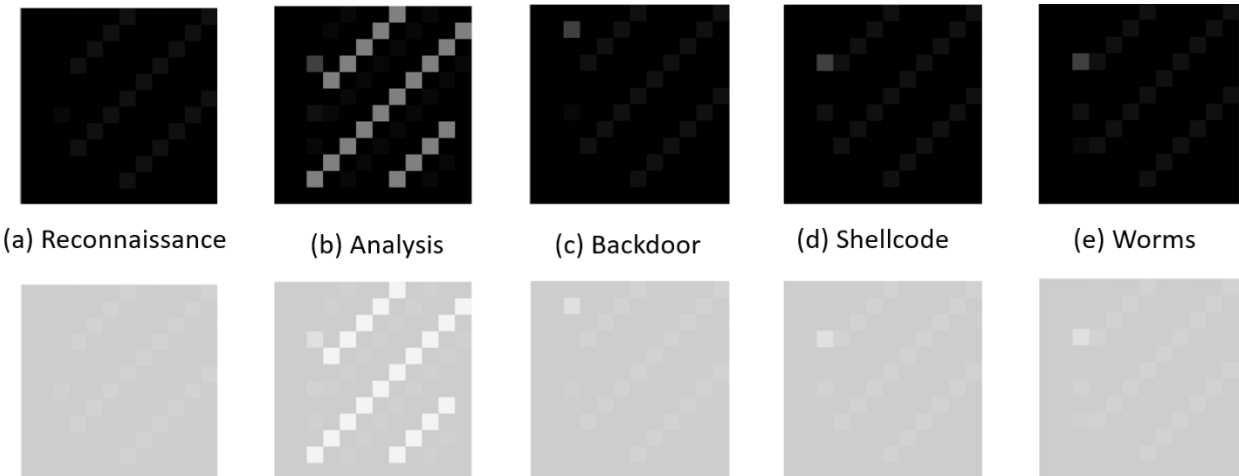


Figure 29: Input image 8x8 of different classes from UNS-NB-15 dataset – Set 2

Injection attack dataset

An injection attack dataset is a time-series numerical data. It is hard to convert into an image. Due to this, the injection attack dataset is converted into the necessary images after certain features extracted from the dataset. More details about the feature extraction can be seen in Section 5.3.3. In total, 11 features will be extracted from the injection attack dataset. These extracted features are

already numeric. The numeric features of the simulated injection attack data have been normalized using the Eq. 5-1. for scaling purpose. "The normalized value has been discretized into 20 intervals using a similar discretizer shown in Fig. 24. The discretizer has a standard scaler of value 0.1 but here it has been used of value 0.05. Hence, it has divided into 20 intervals instead of 10. These 20 intervals have been assigned after on with 20 different bits for binary extraction" [24].

"The simulated sensor data has been converted into 240 bits binary vector by assigning 20 bits to each of 12 extracted features via the discretizer. Now, each 8 bits of 240 bits have been converted into grayscale pixels and then the pixels are reshaped into 6x6 matrix where the vacant six pixels are filled with zeros. In order to create an 8×8 pixel matrix the previous pixel matrix is padded with zeros circularly, as shown in Fig. 30. Finally, the same 8x8 pixel matrix has been sent to the red green and blue channels to represent converted sensor data as images" [24].

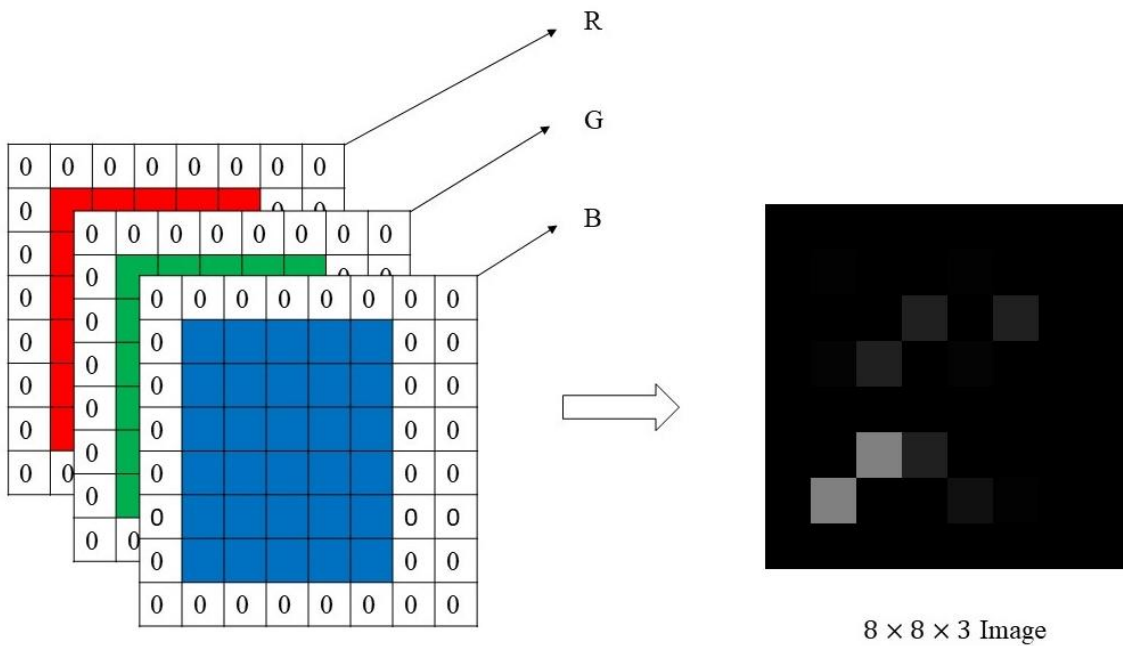


Figure 30: Image representation of injection attack dataset

The generated images for normal and attack types in sensor dataset are represented in Fig. 31 & 32. These images are just a sample from the dataset. A deep insight into the image can show a slight difference between the normal and attack types. These images are fed to CNN for training and later for testing.

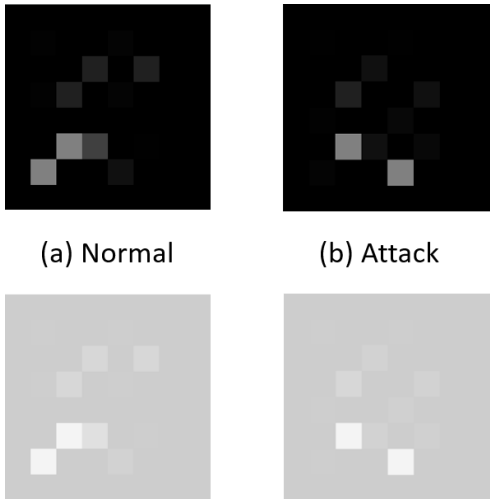


Figure 31: Input image 8x8 of level sensor data

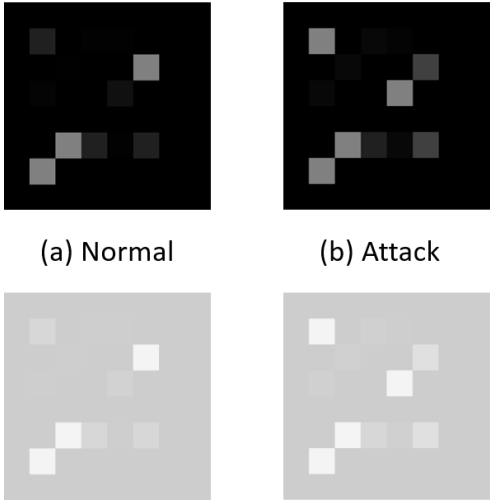


Figure 32: Input image 8x8 of pump data

All the above-mentioned procedures for different datasets to make the dataset suitable for training and testing comes under pre-processing.

5.4. Feature extraction

"Feature extraction is vital for all data driven applications such as pattern recognition and image processing. The derived features out of raw data intend to be more informative and non-redundant facilitating the subsequent learning and generalization steps and in some cases, leading to better

human interpretations. Sometimes some feature extraction techniques are also considered as a data reduction mechanism in pre-processing stage. Different features and features extraction techniques are available. Some common and significant feature categories are statistical features, mathematical features and features extracted through traditional ML techniques" [24].

Statistical features

"Statistics termed as a branch of mathematics dealing with the collection, analysis, interpretation and presentation of masses of numeric data. Statistical features are those which are defined and calculated through statistical analysis. Statistical analysis theory is the frequently used method of data feature extraction in the time domain [155]. It can analyze according to the statistical laws when several objects and several indices are interrelated. Statistical methods are based on forceful theory, have lots of algorithms, and can effectively analyse and process the data. Several statistical factors exist out of which some most commonly used are mean, median, variance, standard deviation, root means square etc. A huge list of statistical features can be found in [156]" [24].

Mathematical features

"Mathematical methods are applied on the raw or pre-processed data to obtain meaningful information. Mathematical features are the most commonly extracted features on both time series and time independent transformations. Several mathematical functions from transform theory can be used to translate the signal into a different domain. List of mathematical features includes derivate, probability and stochastic process, estimation theory, numerical methods etc" [24].

Machine learning based features

"ML techniques are not only used to perform classification or clustering but can be used for feature extraction from raw data and complex data structures. The features such as efficient data compression and data reduction are performed with ML techniques for feature extraction. One good example of machine learning based feature extraction is Principal Component Analysis (PCA). Even deep learning techniques such as autoencoders and Boltzmann machines can also be used for feature extraction. ML techniques are also used to extract features out of features i.e. the extracted features from other techniques or even from ML techniques can be again given to ML based feature extraction to get much refined or complex features" [24].

"The set of features extracted from raw data differs from the network packets and the choice of features extracted such as statistical, mathematical or ML based need to be made based on the application and acquired data" [24].

5.4.1. Feature extraction NSL-KDD dataset

For NSL-KDD dataset, during the preparation of dataset itself, the providers are able to extract such features from the dataset to implicate the influence of the attacks and network behavior into the generated features. NSL-KDD dataset consists of 41 features mentioned in Tables 4-2 to 4-5. As the dataset is already given with features, the pre-processing of the dataset that is suitable for deep learning algorithms covers the feature conversion process for the NSL-KDD dataset. These 41 features are given further to deep learning algorithms for the extraction of complex features from NSL-KDD dataset before attack classification.

5.4.2. Feature extraction UNSW-ND-15 dataset UNSW-NB-15 dataset

Similar to NSL-KDD dataset, the providers of UNSW-NB-15 dataset during the preparation of dataset itself, the providers are able to extract such features from the dataset to implicate the influence of the attacks and network behavior into the generated features. UNSW-NB-15 dataset consists of in total of 47 features mentioned in Tables 4-11 to 4-15. Out of these 47 features, only 43 features were selected during the pre-processing. The feature time stamp and nominal features like IP addresses and port numbers are not useful and even hard to translate them into the numerical format. The pre-processing of the selected features to numerical format and feature selection covers the feature extraction phase for the UNSW-NB-15 dataset. These 43 features are given further to deep learning algorithms for the extraction of complex features from UNSW-NB-15 dataset before attack classification.

5.4.3. Feature extraction injection attack dataset

As raw sensor signal is simply a single numeric value at an instance of time, it is not completely meaningful to feed this data directly to a deep learning algorithm. Therefore, some meaningful features need to be extracted out of the sensors data. As mentioned earlier, we can extract as many features as possible from a dataset. From this huge feature set possible, some statistical and mathematical features were selected to extract from the inject attack data. The selected features are listed in Table 5-3.

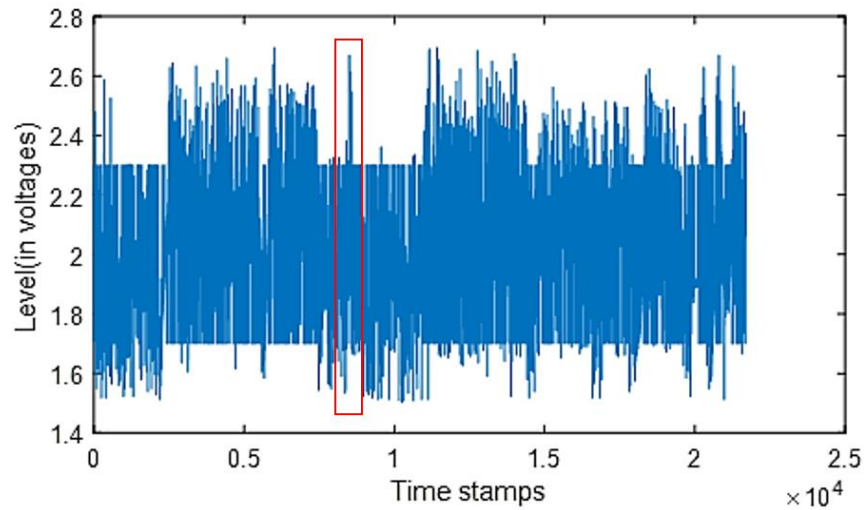


Figure 33: Use of windowing technique for feature extraction from sensor data

In order to extract the necessary features, windowing technique is adapted. There exist several techniques for choice of window depending on the application and on type of data. In our case, the selected sized on window should have either one attack sample or completely without attack. Due to this reason, a relatively small windows size of is selected in an empirical manner for feature extraction. After experimenting with different window sizes, a windows size of 10 samples is selected and slid across the entire sensor data and a sample is illustrated in Fig. 33. From each window, the necessary features are extracted. The extracted features set is mapped to their corresponding labels with normal (0) or attack (1). The labelling is based on the injected attack samples.

Table 5-3: Selected feature for injection attack identification

Feature No.	Feature Name	Feature Type
Feature 1	Mean	Statistical
Feature 2	Standard Deviation	Statistical
Feature 3	Variance	Statistical
Feature 4	Median	Statistical
Feature 5	RMS (Root Mean Square)	Statistical
Feature 6	Maximum Value	Statistical
Feature 7	Minimum Value	Statistical
Feature 8	Peak to RMS	Statistical
Feature 9	Mean of First Differential	Mathematical
Feature 10	Mean of Second Differential	Mathematical
Feature 11	Kurtosis	Statistical

The features extracted from the sensor data are continuous or discrete. Due to this reason, the extracted feature set is normalized. Min-Max normalization is performed to bring different feature values into the range [0, 1]. After the normalization, the data is split into train and test data and the dataset is ready for training the deep learning algorithm.

The features extracted on the level sensor data for measurement injections are illustrated in the following Figures 34, 35 & 36.

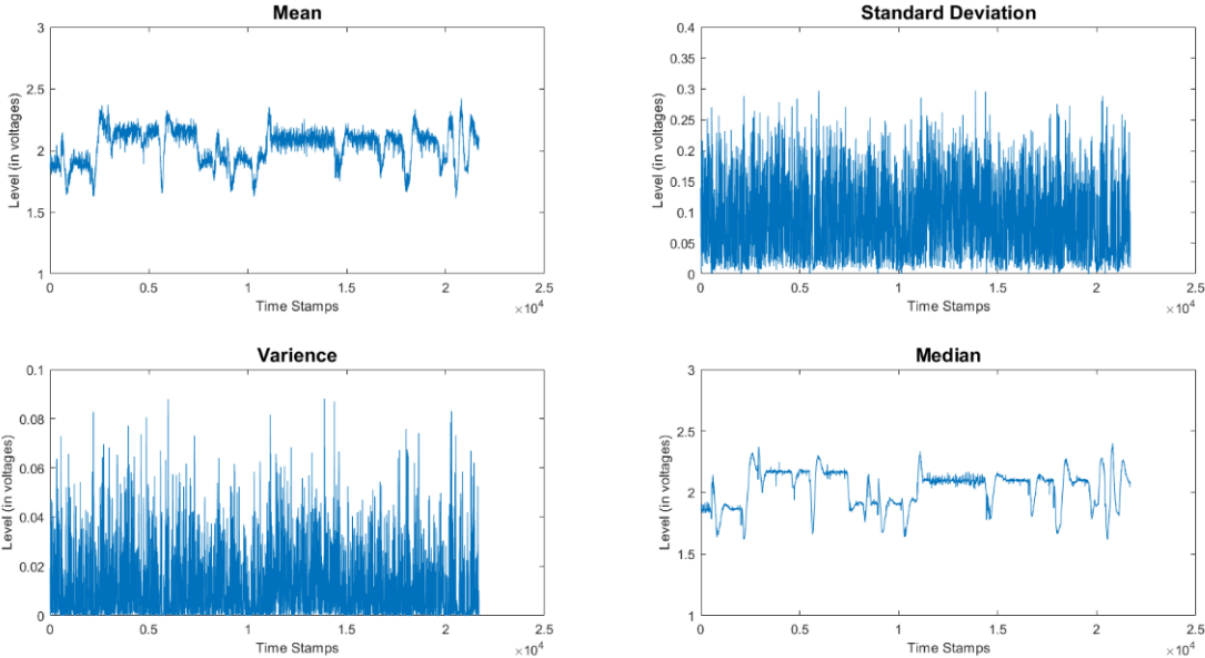


Figure 34: Extracted feature set 1 on level sensor data (a) mean (b) standard deviation (c) variance (d) median

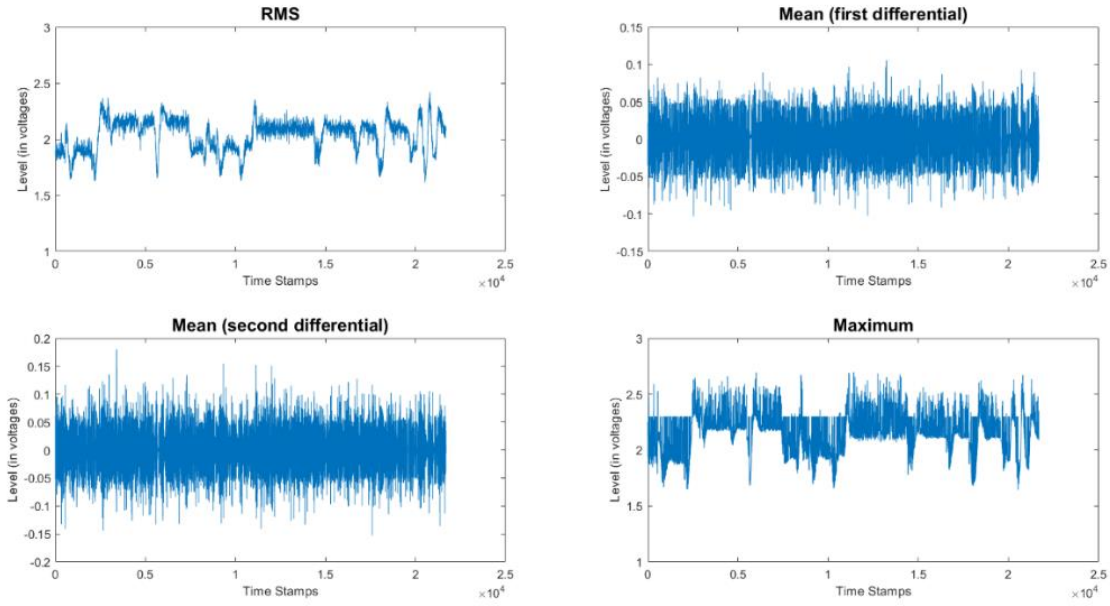


Figure 35: Extracted feature set 2 on level sensor data (a) RMS (b) mean (first differential) (c) mean (second differential) (d) maximum

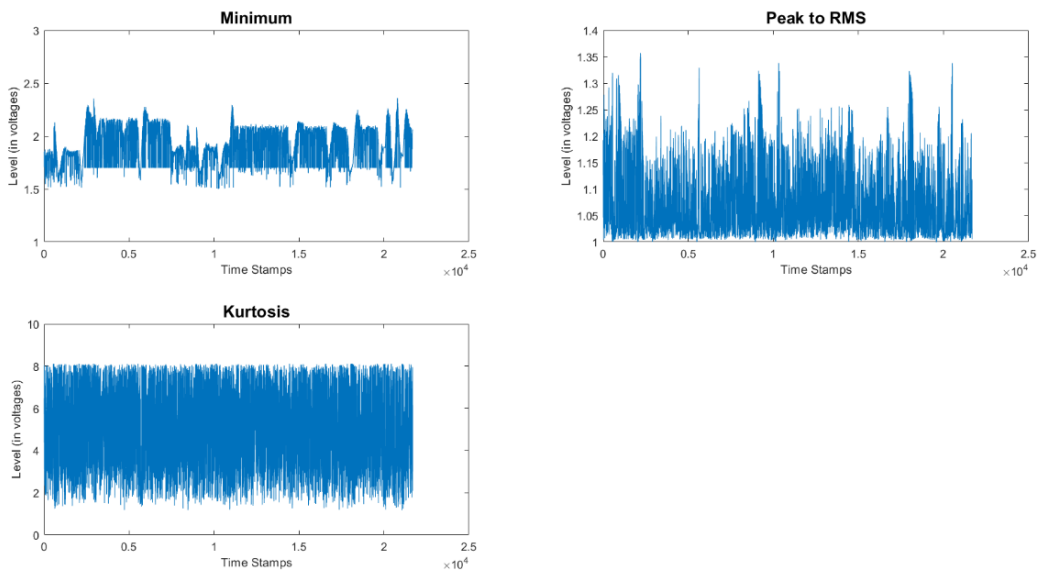


Figure 36: Extracted feature set 3 on level sensor data (a) minimum (b) peak to RMS (c) Kurtosis

The features extracted on the pump data for command injections are illustrated in the following Figures 37, 38 & 39.

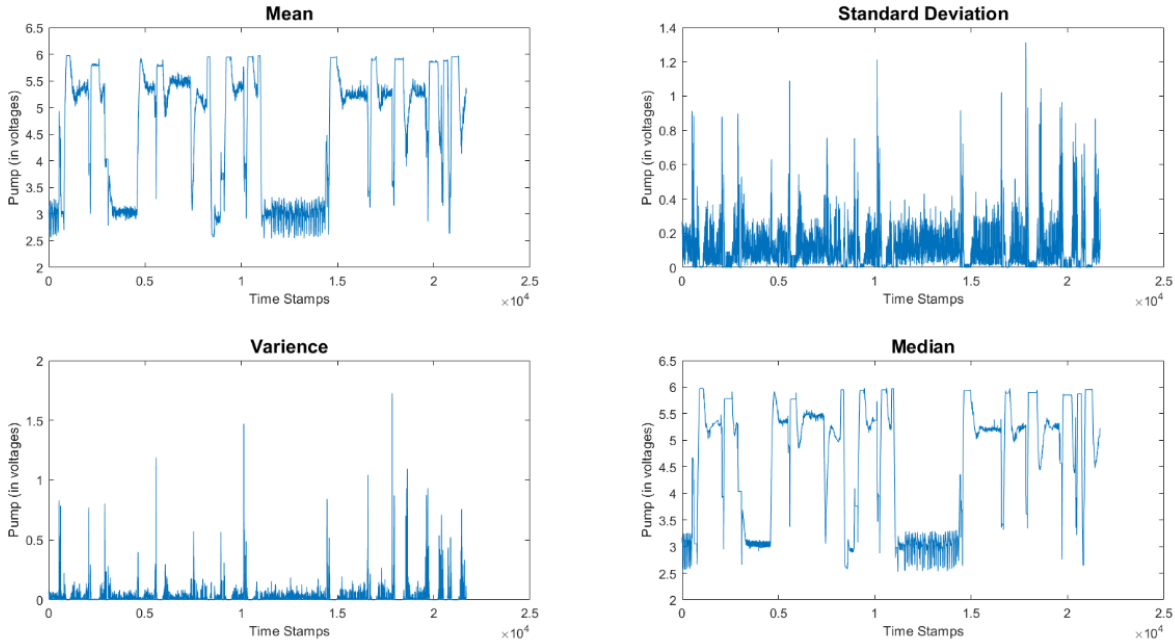


Figure 37: Extracted feature set 1 on pump data (a) mean (b) standard deviation (c) variance (d) median

From the features illustrated, we can see that some additional and multi-variant information is extracted out of the raw sensor data. But for normal observations, it is hard to identify the attacks from those representations. These features set are fed to the deep learning algorithms for further extraction of complex correlations and identify the attacks present in the data.

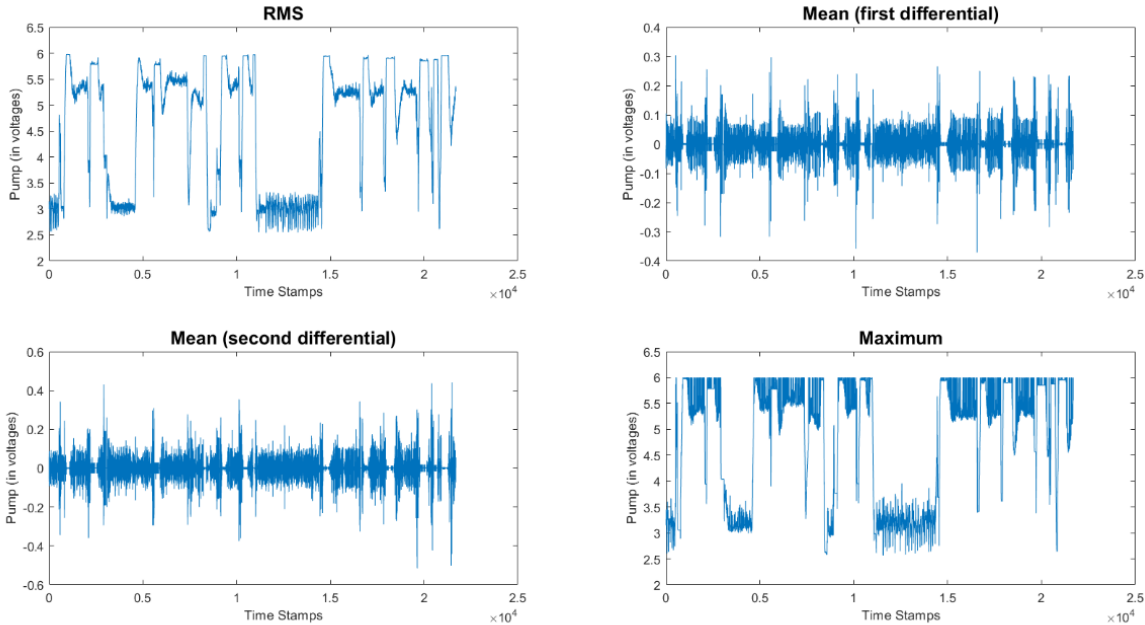


Figure 38: Extracted feature set 2 on pump data (a) RMS (b) mean (first differential) (c) mean (second differential) (d) maximum

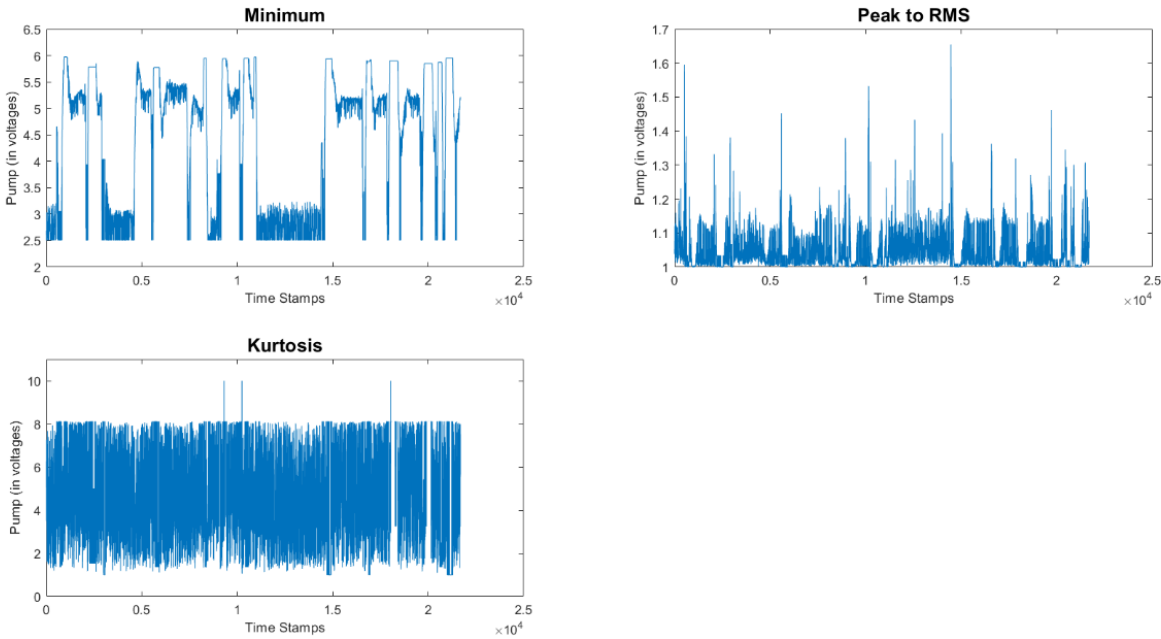


Figure 39: Extracted feature set 3 on pump data (a) minimum (b) peak to RMS (c) Kurtosis

5.5. Concepts of Deep Learning algorithms

"Deep learning techniques are a specific type of machine learning techniques combining both supervised and unsupervised techniques inspired from the human brain. The human brain has got multiple levels of representations with simple features at lower levels and more abstract features at higher levels. Similarly, deep learning consists of multiple hidden layers with initial layers representing the information at lower levels and the final layers representing the information in an abstract format. Some common deep learning architectures include Stacked Auto-Encoders (SAE), Deep Belief Networks (DBN), Convolution Neural Networks (CNN) etc" [113]. For the development of efficient IDS in securing ICS, the traditional deep learning models are used and configured accordingly to the requirements of the application and available datasets so that there are capable of efficiently identifying the attacks. In addition, some theoretical details about different deep learning algorithms used in the scope of this thesis are discussed followed the appropriate use of the algorithms for the desired application.

5.5.1. Stacked Autoencoders

Auto-Encoders (AE) are simple learning architecture which aims to transform inputs into outputs with the least possible amount of distortion. An AE is considered as a feed forward neural network that can learn a compressed, distributed representations of a dataset [157]. Even there are very simple structures, they play an important role in machine learning. Auto-encoders were first introduced in 1980's by Hinton and the PDP group [158] to address the problem of "back propagation".

Stacking the single level AE initializes a deep neural network and works in the same way as in single autoencoders and is considered as a Stacked Autoencoder (SAE). Due to this stacking of AE's, SAE is considered as a deep learning model. SAE uses the same autoencoder principle defined above as a building block to create a Deep Neural Network (DNN) architecture. SAE has many interesting applications such as data compression, visualization etc. Here the output of an individual hidden layer (AE) will be the input to the next hidden layer (another AE). The structure of the stacked autoencoders is shown in Fig. 40.

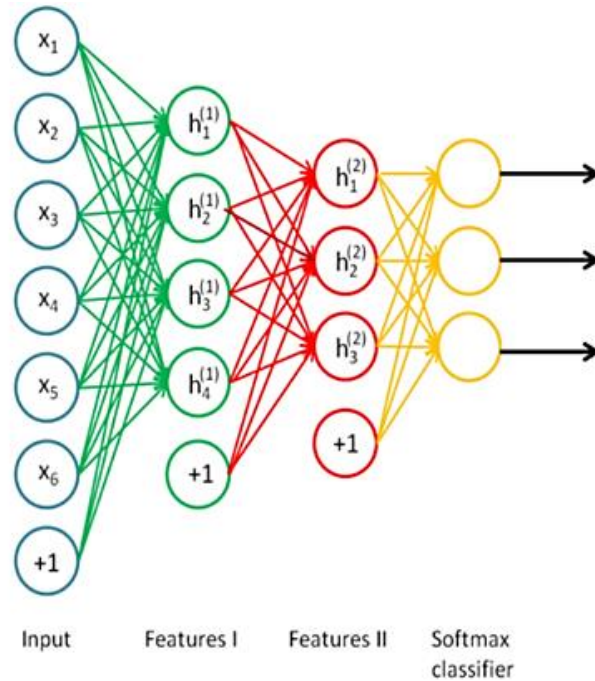


Figure 40 : Structure of stacked autoencoder

Similar to the human brain, the inputs are initially represented in a lower level of abstraction at first AE layer represented in Fig. 40 as Feature 1. These extracted features are fed to the next layer and again more complex features, but a lower number are extracted. In this case with two AE layers, the extracted features are of a higher level of abstraction. Increase in the number of hidden layers increases in higher level of abstraction of the features as well as the inputs.

"The training of stacked autoencoders is done in two phases a pre-training phase and a fine-tuning phase. In the pre-training phase, each layer is considered as a single AE and trained individually in an unsupervised manner. This process involves the feature reduction from the input data. Later, the features extracted from the previous layer are fed to the next layer as inputs for more complex and meaningful feature extraction. This process repeats until the number of autoencoder layers in the stack. The training of stacked autoencoders is done in a greedy layer wise approach. It means each layer is trained at a time. Now a days due to availability of high computation resources at lower prices, for some applications all layers can be trained at a time" [103].

The fine-tuning phase comes after the trained stacked autoencoder was coupled to the classification/regression concept based on the applications need. At this stage, the extracted

features are given as input to supervised learning approaches such as SoftMax regression or SVM or logistic regression etc. for efficient learning mechanism yielding a supervised deep neural network architecture. The fine-tuning phase comes after the trained stacked autoencoder was coupled to the classification mechanism. This fine-tuning process of the algorithm is usually a backpropagation algorithm. This improves the performance of the autoencoder by adapting its weights.

5.5.2. Deep Belief Networks

"Deep Belief Networks (DBN) are formed by stacking Restricted Boltzmann Machines (RBM). RBM is a generative stochastic network which can learn a probability distribution over its set of inputs [159]. The main difference of RBM to a feed forward neural network was, the connection between the visible and hidden layers are undirected. That means, the values can propagate from visible layer to hidden layer and from hidden layer to visible layer. But RBM, in contrast, don't have any connection between the neurons of visible layers and hidden layers like a normal Boltzmann machine" [103].

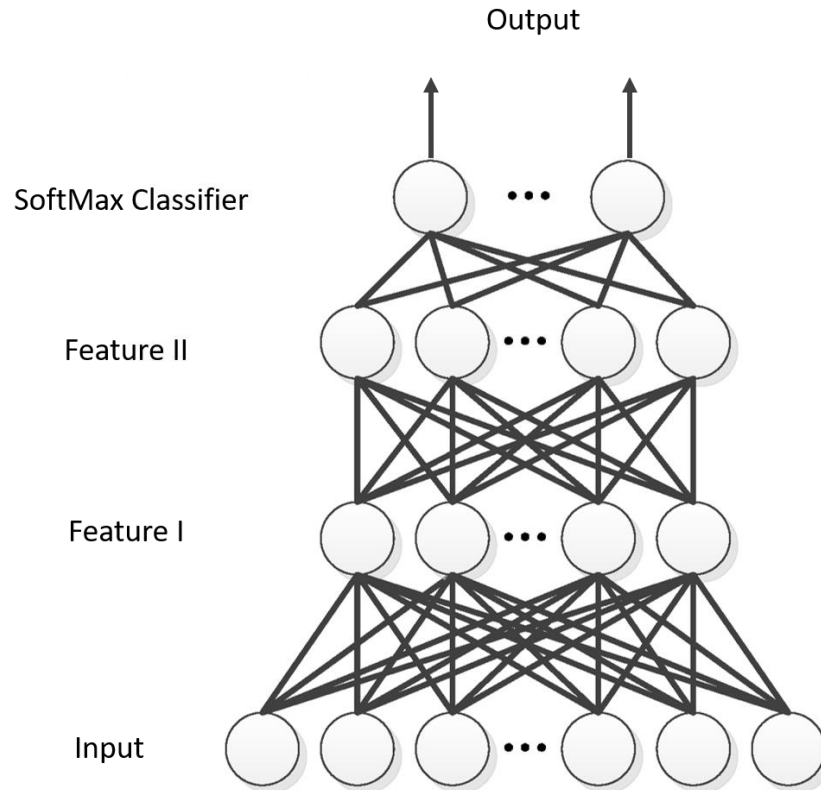


Figure 41: Structure of deep belief network

"The main objective of using this algorithm is to bring the recreated data to be as close as the real-world data, and this is achieved by using the weight update method. Stacking of the RBMs forms a DBN. As mentioned for stacked autoencoders, DBN will also have pre-training phase done in greedy wisest approach and the fine-tuning process is done after coupling with other machine learning algorithm for classifying attacks" [103].

Fig. 41 shows the architecture of the stacked version of RBM termed as DBN. The key difference of AE to RBN is, in AE the Euclidean cost function between the input and the output of the AE is minimized whereas in RBN, the network weights adapted in such a way that probability of representation of the input to the output is maximized. In simple terms, AE / SAE uses mathematical models for network weight adaption whereas RBN / DBN uses probabilistic models for network weight update mechanisms. A more detailed information about DBN can be found at [160].

5.5.3. Convolutional Neural Networks

Convolutional Neural Networks (CNN) is another category of deep learning algorithm and are considered as an extension to the traditional feed forward neural networks. CNN have proven very effective in many application domains such as image recognition and classification, speech processing applications etc. Its effectiveness has been successfully proven in tasks such as identifying faces, objects and traffic sign detection mainly used in robotics and self-driving cars.

Four main operations of CNN comprise of Convolution layer, non-linear activation function such as ReLU, pooling layer, and fully connected layer (classification). Combining the above-mentioned key parameters forms the CNN. The convolution and pooling layers act as a feature extraction mechanism out of an image while the fully connected layer act as a classifier. More detailed discussion on CNN is discussed in [161]. Fig. 42 will give an overview of the CNN.

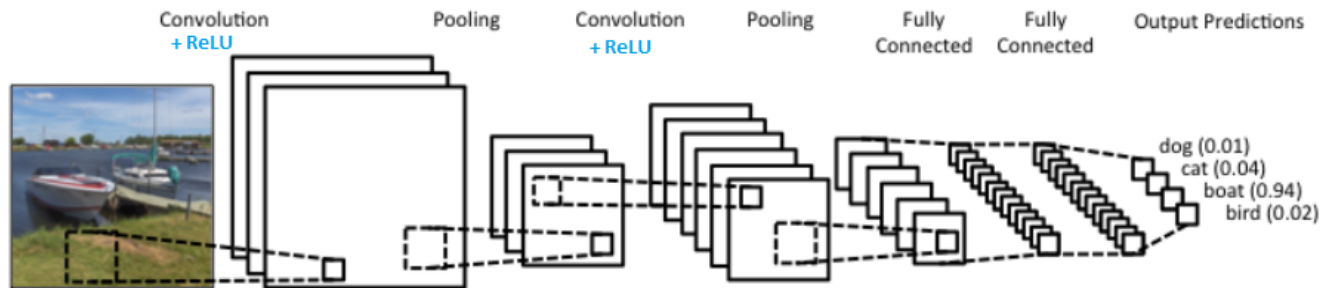


Figure 42. Structure of convolutional neural network

5.5.4. Algorithm configuration parameters

a. Weight initialization

Training of any DNN in general starts with random weights [162] [163]. These random weights are normally chosen within a range of $(-1,1)$. However, these random initializations can produce a set of weights that are sometimes hard for backpropagation to train. Some researchers have proposed some weight initialization algorithms that provide a good set of starting weights for backpropagation [164]. [165] introduced weight algorithms termed as Xavier weight initialization algorithm. Because of its ability to produce consistently performing weights which are suitable for

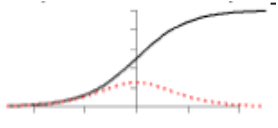
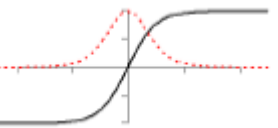
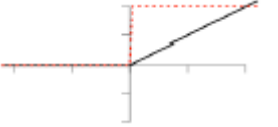

backpropagation, this technique becomes rather popular. In this thesis, the performance of this initialization was also evaluated.

b. Transfer functions for deep learning

The backpropagation of DNN also relies on the derivatives of the transfer functions to incrementally calculate or propagate, error corrections from the output neurons back through the weights of the network. Before 2011, most of the neurons in the hidden layers used the hyperbolic tangent or the logistic transfer function, which are the sigmoid functions as the transfer functions. The derivative of both of their functions saturates to 0 as the input approaches, either positive or negative infinity, which causes this transfer function to exhibit the vanishing gradient problem [166]. [167] introduces the ReLU transfer functions to address this problem.

Usually, in image processing applications, ReLU transfer function usually achieves better training results for DNN than the sigmoid transfer function. In modern DNN, the type of transfer function to be used can be defined separately for each individual hidden layer. For the feature extracting hidden layers of DNN, either sigmoid or ReLU transfer functions can be used based on the application. For the output layer, if it is regression linear transfer function is used and if it is classification SoftMax transfer function is used. No transfer function is necessary for the input layer. Following Table. 5-4 summarizes the logistic, hyperbolic tangent, ReLU, linear, and SoftMax transfer functions:

Table 5-4: List of transfer functions commonly used in deep learning

Name / Range	Expression(Forward)	Derivative (Backward)	Graph (Derivatives in Red)
Logistic / Sigmoid [0, 1]	$\Phi(x) = \frac{1}{1 + e^{-x}}$	$\Phi'(x) = \frac{e^x}{(1 + e^{-x})^2}$	
HTan [-1, 1]	$\Phi(x) = \text{htan}(x)$	$\Phi'(x) = 1 - \Phi^2(x)$	
ReLU [0, +∞]	$\Phi(x) = \max(0, x)$	$\Phi'(x) = \begin{cases} x > 0 & 1 \\ x \leq 0 & 0 \end{cases}$	
Linear [-∞, +∞]	$\Phi(x) = x$	$\Phi'(x) = 1$	
SoftMax [-∞, +∞]	$\Phi(x) = \frac{1e^{x_j}}{\sum_{k=1}^{ x } e^{x_k}}$	NA	NA

The graph column of the table shows the transfer function and the derivative of that transfer function. The solid black line is the transfer function output and the dotted red line is the derivative.

c. Regularization

Overfitting and underfitting are most common problems while using the ML algorithms. When coming to DNN, overfitting is a frequent problem [168]. Regularization is a concept designed to combat overfitting. One most common form of regularization is to simply add a scaled summation

of the weights of the neural network to the loss function. This calculation will cause the training algorithm attempts to lower the weights of the neural network along with the output error. The most common forms of this weight regularization are L1 and L2 [169].

L1 regularization sums the weights of the neural network (w) and produces an error value (E_1) that is added to the loss function of the network. The equation is shown in the following equation

$$E_1 = \frac{\lambda_1}{n} \sum_w |w| \quad (5-2)$$

The w vector includes only the actual weights but not the bias weights. The value λ_1 is a scaling factor for the effect of the L1 regularization. If λ_1 is too high, the objective of lowering the weights will overwhelm the once for achieving a lower error for the neural network training. This situation causes a failure of the neural network to converge to a lower error. The value n represents the number of training set elements.

L2 regularization is defined as similarity to L1 and is provided by the following equation.

$$E_2 = \frac{\lambda_2}{n} \sum_w w^2 \quad (5-3)$$

Both L1 and L2 regularization sum the weights without regard to their sign. This magnitude-oriented approach is accomplished by an absolute value for L1 and a square of L2. The weights are pushed toward 0 in both cases. However, L1 has a greater likelihood of pushing the weights entirely to 0 and effectively pruning the weighted connection [169]. This pruning feature of L1 is especially valuable in this research as it works as a function for efficient feature selection. L1 will specify the worthless engineered features by pruning them.

Apart from L1 and L2 regularization, [170] introduced dropout as simple regularization technique for DNN. Dropout is typically implemented as a single dropout layer.

d. Batch size

There are three different techniques to decide how often the weights are updated. They are online, full batch and mini batch [171].

- Online learning: Updates weights after each training data instance; thus, it takes more computation time to complete the learning compared to other techniques.

- Full batch: It runs a full sweep through the training data and updates weights. However, it is impractical to run full batch learning for a big dataset such as 60,000+ training samples.
- Min-batch: It divides a dataset into small chunks of data and performs the learning for each chunk. This method allows matrix-matrix multiples in software programming and takes less computation time and are more efficient on GPU's.

e. Enhancements to backpropagation

Several techniques exist for the enhancements apart from basic backpropagation and weight update rule. Momentum has been a significant component of backpropagation training for some time. [172] introduced the seminal momentum algorithm that is a regularization technique for gradient ascent/descent. Momentum backpropagation influences the current iteration's weights by adding a portion of weight change from the previous iteration.

$$v_t = \gamma v_t - \eta \nabla_{\theta_{t-1}} J(\theta_{t-1}) \quad (5-4)$$

Accordingly, the weight updates have the necessary momentum to push through local minima and to continue the descent of the output of the loss function.

[173] improves the momentum calculation by increasing the effectiveness of random mini-batches selected by SGD termed as Nesterov momentum. This decreases the likelihood of a particular bad mini-batch from changing the weights into an irreparable state. A NN update rule using the Nesterov momentum was introduced by [174].

Several researchers also developed different update rules beyond the classic backpropagation and Nesterov momentum update rules. The user needs to choose learning rate and momentum training parameters that are applied across all weights in NN using either classic backpropagation rules or with Nesterov momentum. Usually, the decay in learning rate as the NN trains is advantageous and proposed by [175]. In addition, sometimes each weight in NN with a different learning rate might be beneficial [176]. [177] introduced the Adaptive Gradient Algorithm (AGA) to decay both the learning rate as well as vary the rate per weight. [178] introduced AdaDelta update rule that uses a window of gradients that determine the learning rate to mitigate the aggressive monotonic learning rate decay of AdaGrad. [179] proposed a separate RMSprop to address the aggressive learning rate

decay of AGA. In this algorithm, the learning rate is determined by dividing gradients by a root means square of the weights.

[180] proposed the Adam update rules, the name is derived from the adaptive moment estimates that it uses. Adam determines the weight corrections by estimating the first (mean) and second (variance) moments. Adam begins with the exponentially decaying averages of past gradients (m):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (5-5)$$

This average achieves a similar goal like classic momentum update, but its value is calculated automatically based in the current gradient (g_t). The update rule then calculates the second momentum (v_t):

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (5-6)$$

The values m_t and v_t are estimates of the first moment (the mean) and the second moment (the un-centered variance) of the gradients. However, a strong bias towards zero during the initial training cycles will be there. The first moment's bias is corrected as follows:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (5-7)$$

Similarly, the second moment is also corrected:

$$\widehat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (5-8)$$

These bias-corrected first and second moment estimates are applied to the ultimate Adam update rule, as follows:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \widehat{m}_t \quad (5-9)$$

This dissertation featured the Adam update rule for all neural network training due to the rule's robustness to initial learning rate (η) and other training parameters. Kingma and Ba (2014) propose default values of 0.9 for β_1 , 0.999 for β_2 , and 10^{-8} for ϵ , that were used in this dissertation.

In addition to the above mentioned deep learning algorithms, researchers have also utilized deep learning for recurrent neural networks. The research community has recently shown considerable interest in deep LSTM networks. [181] used a grid of LSTM units to achieve greater accuracy.

[182] introduced the gated recurrent network (GRU) and added an output gate, which allows greater accuracy as the time series increases in length. Unlike feedforward neural networks, LSTM and GRU are recurrent networks that can function as Turing machines [183].

5.6. Usage of Deep Learning algorithms

The proposed deep learning algorithms are the basic methods or model that can be used for any application. But it needs to be adapted the necessity of the individual application. This sub-section discussed the proper implementation of the mentioned deep learning algorithms such that an efficient security mechanism can be established using those algorithms. The implementation of the individual deep learning algorithms needs to be discussed in accordance with their datasets.

5.6.1. NSL-KDD dataset - Stacked Autoencoder

The deep learning model of implementing the SAE on NSL-KDD dataset is represented in Fig. 43. From Fig. 43. we can see that the input layer with 41 nodes which represent the 41 features of NSL-KDD dataset. The hidden layers 1,2,3 are used for complex feature extraction. During pre-training hidden layer 1 reduces the input features from 41 to 30 followed by hidden layer 2 which reduces the 30 features to 20 Features followed by hidden layer 3 that reduces the 20 features further to 10 features. Due to this reason, hidden layer 1 have 30 neurons, hidden layer 2 have 20 neurons and hidden layer 1 have 10 neurons. The output of the third hidden layers are considered as extracted features from NSL-KDD dataset.

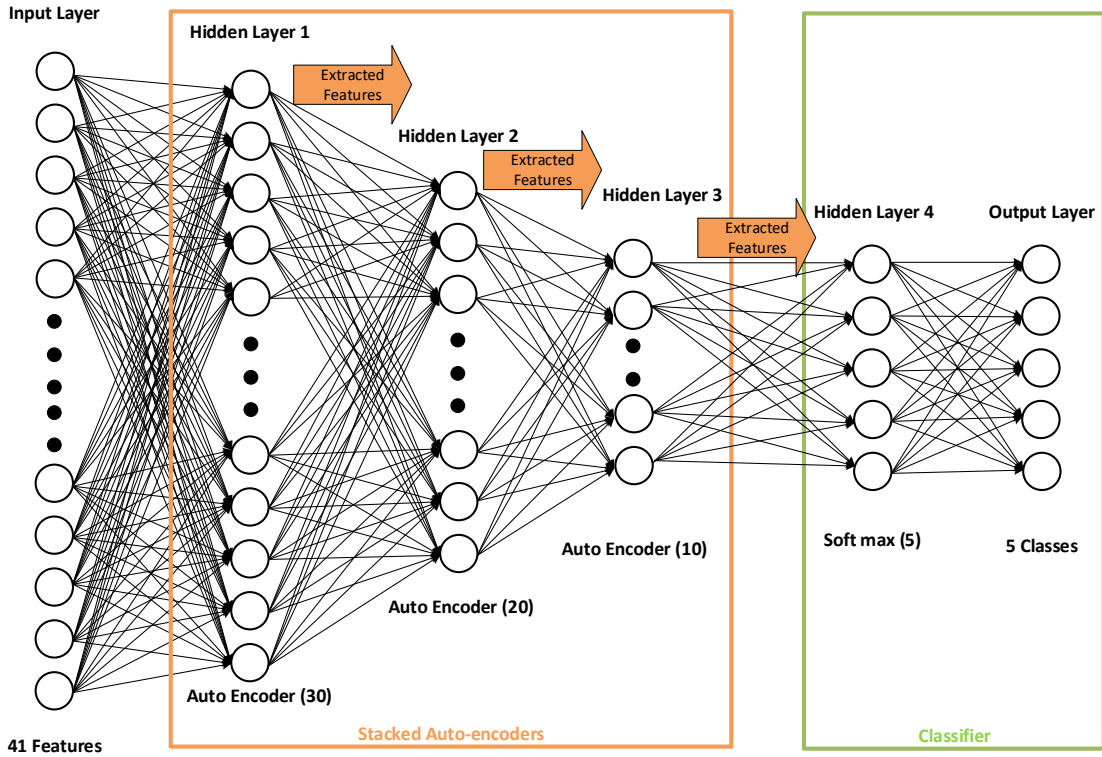


Figure 43: Implementation of SAE on NSL_KDD dataset

The features extracted during the pre-training phase are purely unsupervised learning. The extracted features are given as an input to the SoftMax Regression (SMR) layer, which is the hidden layer 4. The SoftMax layer will use the 10 features and identify the normal and attack classes based on the requirements. The no. of output classes is based on the requirements according to the Table 5-6. The SoftMax layer is trained in a supervised manner with the extracted features and labels termed as fine-tuning step 1. Now, the whole SAE with all hidden layers (1-4) is fine-tuned via backpropagation in a supervised manner. This procedure improves the extracted features for the learning of the SAE. This process is considered as fine-tuning step 2. After the process of fine-tuning, the trained network is validated with validation dataset. The validation of the training is performed on the entire network. Validation prevents the network from overfitting. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.2. NSL-KDD dataset - Deep Belief Network

The deep learning model of implementing the DBN on NSL-KDD dataset is represented in Fig. 44. From Fig. 44, we can see that the input layer with 41 nodes which represent the 41 features of NSL-KDD dataset. The hidden layers 1,2,3 are used for complex feature extraction. During pre-training hidden layer 1 reduces the input features from 41 to 30 followed by hidden layer 2 which reduces the 30 features to 20 Features followed by hidden layer 3 that reduces the 20 features further to 10 features. Due to this reason, hidden layer 1 have 30 neurons, hidden layer 2 have 20 neurons and hidden layer 3 have 10 neurons. The output of the third hidden layers are considered as extracted features.

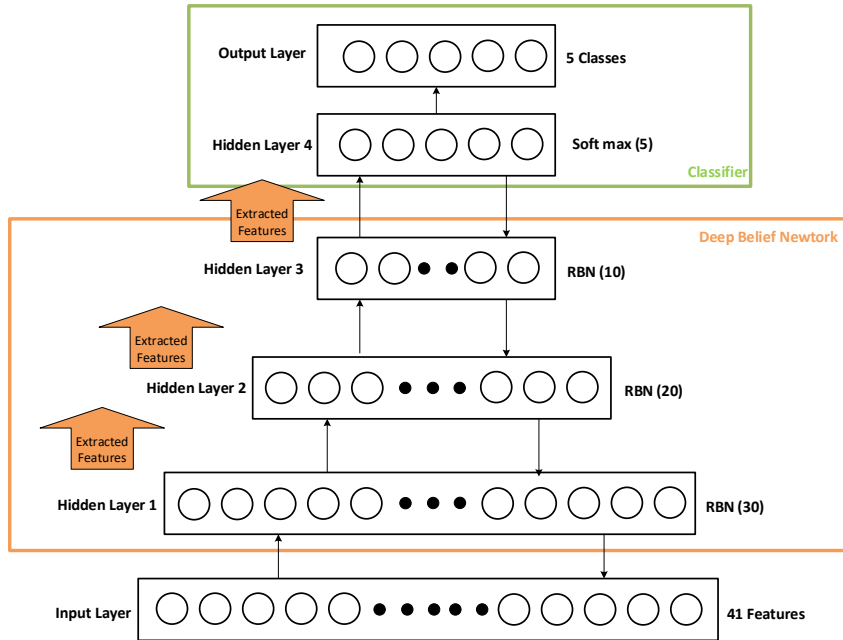


Figure 44: Implementation of DBN on NSL_KDD dataset

The features extracted during the pre-training phase are purely unsupervised learning. The extracted features are given as an input to the SMR layer which is the hidden layer 4. The SoftMax layer will use the 10 features and identify the normal and attack classes based on the requirements. The no. of output classes is based on the requirements according to the Table 5-6. The SoftMax layer is trained in a supervised manner with the extracted features and labels termed as fine-tuning step 1. Now, the whole DBN with all hidden layers (1-4) is fine-tuned via back propagation in supervised manner. This procedure improves the extracted features for learning of the DBN. This process is considered as fine-tuning step 2. After the process of fine-tuning, the trained network is

validated with validation dataset. The validation of the training is performed on the entire network. Validation prevents the network from overfitting. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.3. NSL-KDD dataset - Convolutional Neural Networks

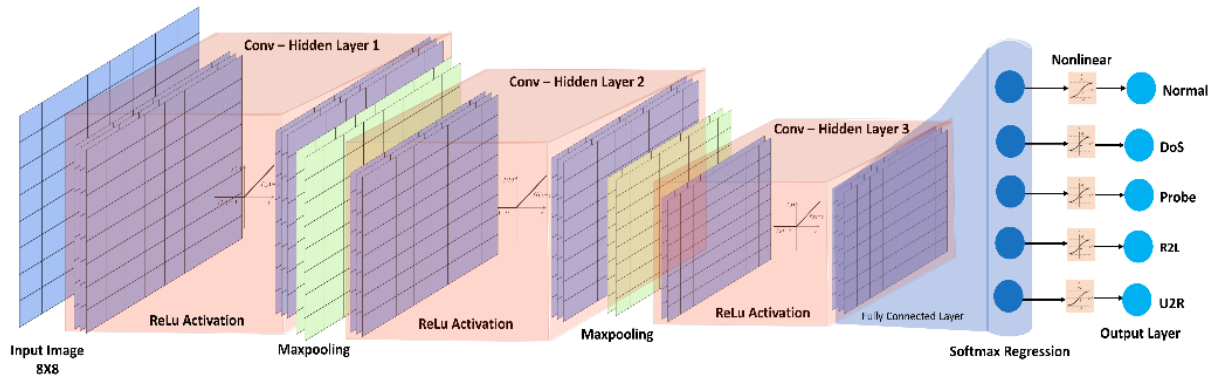


Figure 45: Implementation of CNN on NSL-KDD dataset

From Fig. 45, we can see the implemented CNN model. It includes the following steps:

- Step 1: The image data set of NSL-KDD dataset obtained from image representation stage with each image of size 8x8 is given as an input image to CNN – Hidden Layer 1.
- Step 2: The CNN layers are initially initialized with Xavier weights and random filters and these are adopted during the training process.
- Step 3: The network takes the input image and initiates the training process. The image goes through the forward propagation steps (convolution, ReLU and pooling operations along with forward propagation of the fully connected layers) and finds the output probabilities.
- Step 4: The error value of the desired output to the generated output is calculated. And validation is performed after every 300 iterations.
- Step 5: Now backpropagation with gradient descent is used to update the network weights and all filter values to minimize the output error.

The above steps are continued until the validation function measures the same value for five times as the patience was set to 5. This ensures the network from overfitting. Narrow convolution technique is used for the first convolution hidden layer. The output feature map of the first convolution hidden layer is smaller than 8x8. Due to this reason, in second and third hidden layers wide convolution technique is used by padding the feature maps with zeros. SoftMax regression with non-linear sigmoid transfer function is used for classification of attack classes at the final fully connected layer. The no. of output classes is based on the requirements according to the Table 5-6. The SMR layer is trained in a supervised manner with the extracted features and labels. The output of the trained CNN for multiple attacks classes present in the dataset.

The validation of results has been done following three facts for all the proposed IDS models and these facts are described using the training and validation curve in Fig. 46. The figure has been chosen for the trained IDS on NSL-KDD dataset for discussing the validation method generally. Validations are done following the exact same procedure for all proposed models.

- i. **Overfitting:** By overfitting, it is ensured that the training model is not memorizing but actually learning something from the given input. This means the difference in between validation accuracy and the training accuracy must be closer to zero. In Fig. 53. the dotted black line is represented for the training accuracy and loss validation and the blue and red lines are represented for actual training accuracy and loss respectively. From the figure it visible, there is no difference in between the training and validation curve for both accuracy and loss. Therefore, it is ensured that all the trained network in this paper is not overfitted.
- ii. **Under-fitting:** By under-fitting it is made certain that the training model or network is not showing immature behaviour. This means the training model is performed well on the training data. The training curve of accuracy and loss in Fig. 53. has proven that the trained models are not suffered from under-fitting because the calculated accuracy on training data is started from 54 % approximately and has gradually increased as the training iteration has progressed as well as the training loss has gradually decreased and this is known as the normal behaviour of a training process.
- iii. **Variance:** By variance, it is verified that the proposed models are trained properly right from the beginning till the end of the training process. This means the validation accuracy

and loss are always closer to the training accuracy and loss throughout the whole training process. No variance has been experienced on both validation and training curve in Fig. 53.

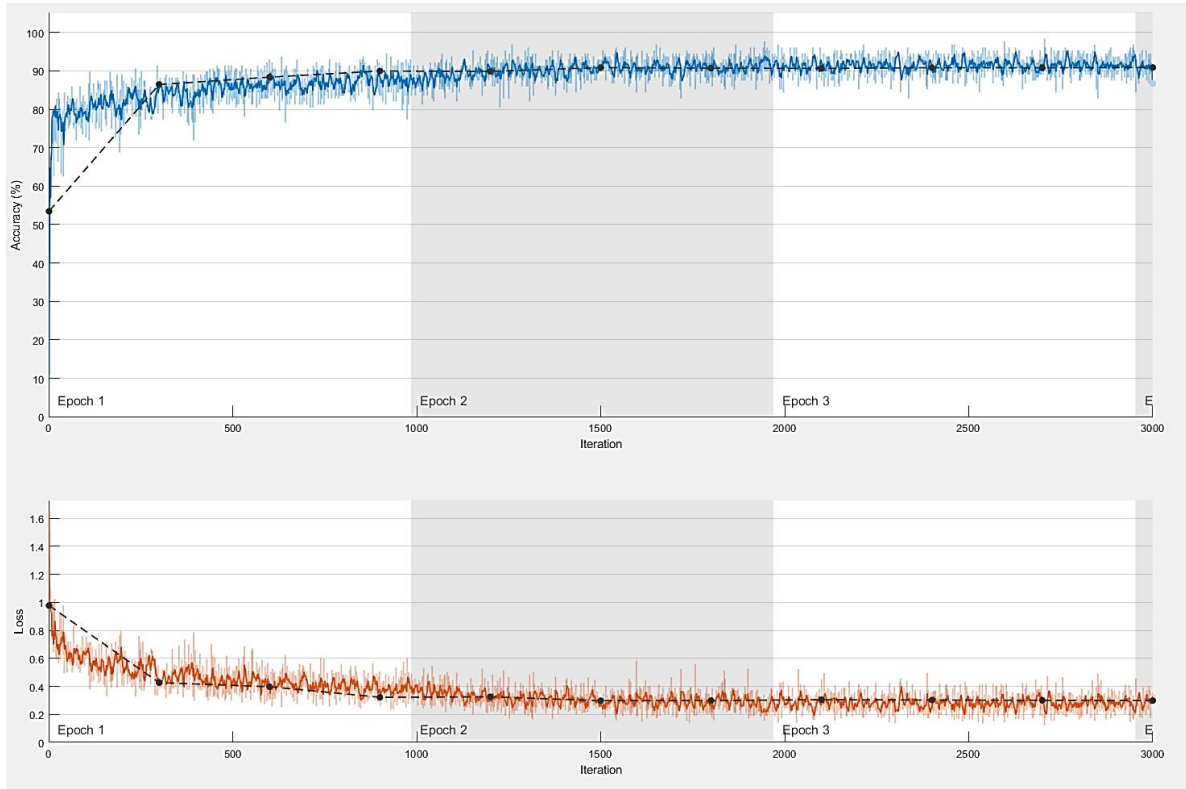


Figure 46: Training and validation curve of CNN for a given dataset

By maintaining these three facts, it is confirmed that the calculated test results are correct and meaningful. The best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.4. UNSW-NB-15 - Stacked Autoencoder

The deep learning model of implementing the SAE on UNSW-NB-15 dataset is represented in Fig. 47. From Fig, 47. we can see that the input layer with 43 nodes which represent the 43 features of UNSW-NB-15 dataset. The hidden layers 1,2,3 are used for complex feature extraction. During pre-training hidden layer 1 reduces the input features from 43 to 30 followed by hidden layer 2 which reduces the 30 features to 20 Features followed by hidden layer 3 that reduces the 20 features further to 10 features. Due to this reason, hidden layer 1 have 30 neurons, hidden layer 2 have 20

neurons and hidden layer 3 have 10 neurons. The output of the third hidden layers are considered as extracted features.

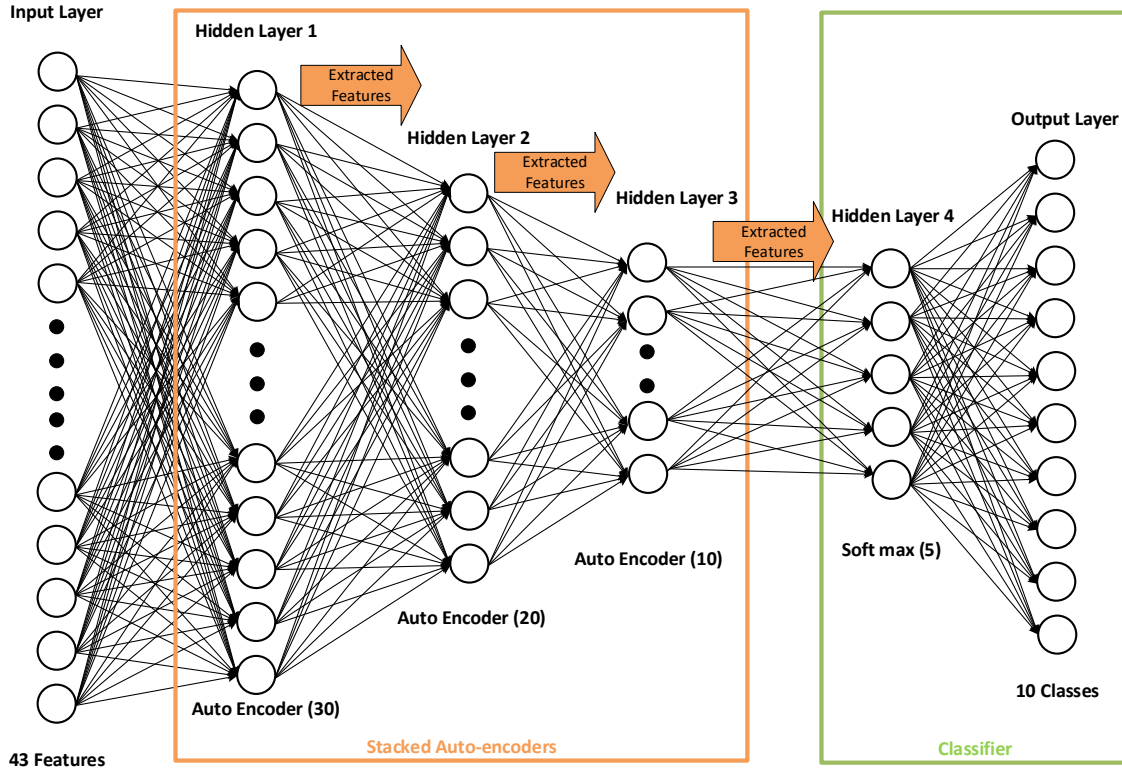


Figure 47: Implementation of SAE on UNSW-NB-15 dataset

The features extracted during the pre-training phase are purely unsupervised learning. The extracted features are given as an input to the SMR layer which is the hidden layer 4. The SoftMax layer will use the 10 features and identify the normal and attack classes based on the requirements. The no. of output classes is based on the requirements according to the Table 5-7. The SoftMax layer is trained in a supervised manner with the extracted features and labels termed as fine-tuning step 1. Now, the whole SAE with all hidden layers (1-4) is fine-tuned via back propagation in supervised manner. This procedure improves the extracted features for learning of the SAE. This process is considered as fine-tuning step 2. After the process of fine-tuning, the trained network is validated with validation dataset. The validation of the training is performed on the entire network. Validation prevents the network from overfitting. Later the best model is tested with the test data

for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.5. UNSW-NB-15 - Deep Belief Networks

The deep learning model of implementing the DBN on UNSW-NB15 dataset is represented in Fig. 48. From Fig. 48, we can see that the input layer with 43 nodes which represent the 43 features of UNS-NB-15 dataset. The hidden layers 1,2,3 are used for complex feature extraction. During pre-training hidden layer 1 reduces the input features from 43 to 30 followed by hidden layer 2 which reduces the 30 features to 20 Features followed by hidden layer 3 that reduces the 20 features further to 10 features. Due to this reason, hidden layer 1 have 30 neurons, hidden layer 2 have 20 neurons and hidden layer 1 have 10 neurons. The output of the third hidden layers are considered as extracted features.

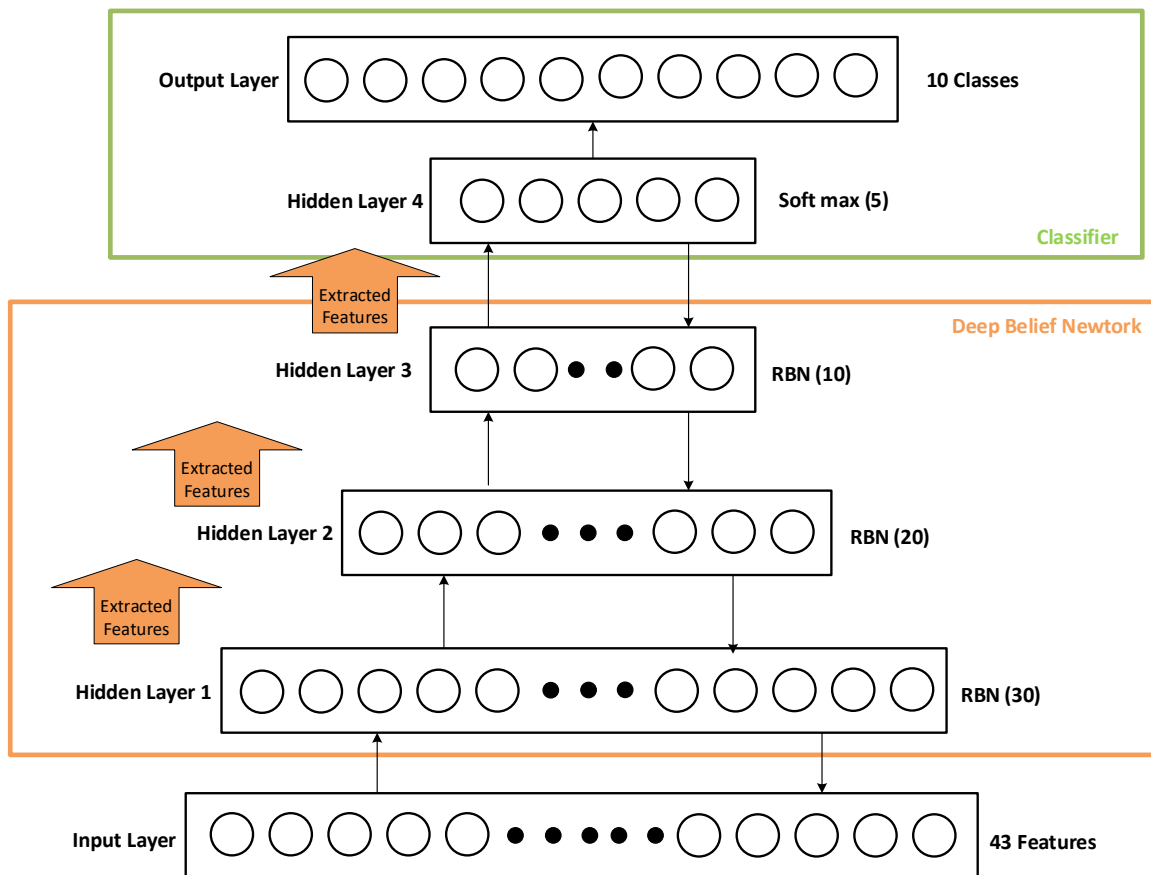


Figure 48: Implementation of DBN on UNSW-NB-15 dataset

The features extracted during the pre-training phase are purely unsupervised learning. The extracted features are given as an input to the SMR layer which is the hidden layer 4. The SoftMax layer will use the 10 features and identify the normal and attack classes based on the requirements. The no. of output classes is based on the requirements according to the Table 5-7. The SoftMax layer is trained in a supervised manner with the extracted features and labels termed as fine-tuning step 1. Now, the whole DBN with all hidden layers (1-4) is fine-tuned via backpropagation in supervised manner. This procedure improves the extracted features for the learning of the DBN. This process is considered as fine-tuning step 2. After the process of fine-tuning, the trained network is validated with validation dataset. The validation of the training is performed on the entire network. Validation prevents the network from overfitting. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.6. UNSW-NB-15 - Convolutional Neural Networks

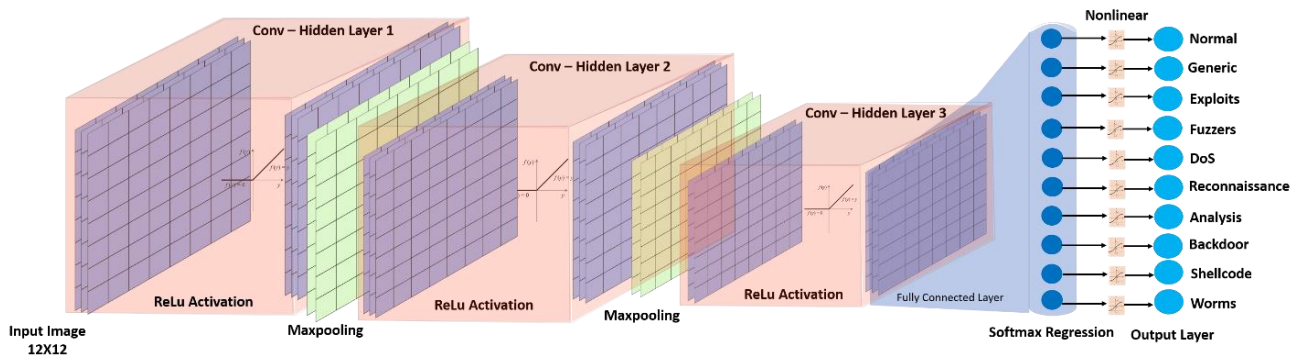


Figure 49: Implementation of CNN on UNSW-NB-15 dataset

From Fig. 49, we can see the implemented CNN model. It includes the following steps:

Step 1: The image data set of NSL-KDD dataset obtained from image representation stage with each image of size 8x8 is given as an input image to CNN – Hidden Layer 1.

Step 2: The CNN layers are initially initialized with Xavier weights and random filters and these are adopted during the training process.

Step 3: The network takes the input image and initiates the training process. The image goes through the forward propagation steps (convolution, ReLU and pooling operations along with forward propagation of the fully connected layers) and finds the output probabilities.

Step 4: The error value of the desired output to the generated output is calculated. And validation is performed after every 300 iterations.

Step 5: Now backpropagation with gradient descent is used to update the network weights and all filter values to minimize the output error.

The above steps are continued until the validation function measures the same value for five times as the patience was set to 5. This ensures the network from overfitting. Narrow convolution technique is used for the first convolution hidden layer. The output feature map of the first convolution hidden layer is smaller than 12x12. Due to this reason, in second and third hidden layers wide convolution technique is used by padding the feature maps with zeros. SoftMax regression with non-linear sigmoid transfer function is used for classification of attack classes at the final fully connected layer. The no. of output classes is based on the requirements according to the Table 5-7. The SMR layer is trained in a supervised manner with the extracted features and labels. The output of the trained CNN for multiple attacks classes present in the dataset. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.7. Injection attack - Stacked Autoencoder

The deep learning model of implementing the SAE on the extracted feature dataset is represented in Fig. 50. From Fig. 50, we can see that the input layer with 11 nodes which represent the extracted features of the raw sensor data. The hidden layers 1,2,3 are used for complex feature extraction. During pre-training hidden layer 1 reduces the input features from 11 to 8 followed by hidden layer 2 which reduces the 8 features to 6 Features followed by hidden layer 3 that reduces the 6 features further to 4 features. Due to this reason, hidden layer 1 have 8 neurons, hidden layer 2 have 6 neurons and hidden layer 3 have 4 neurons. The output of the third hidden layers are considered as the extracted features.

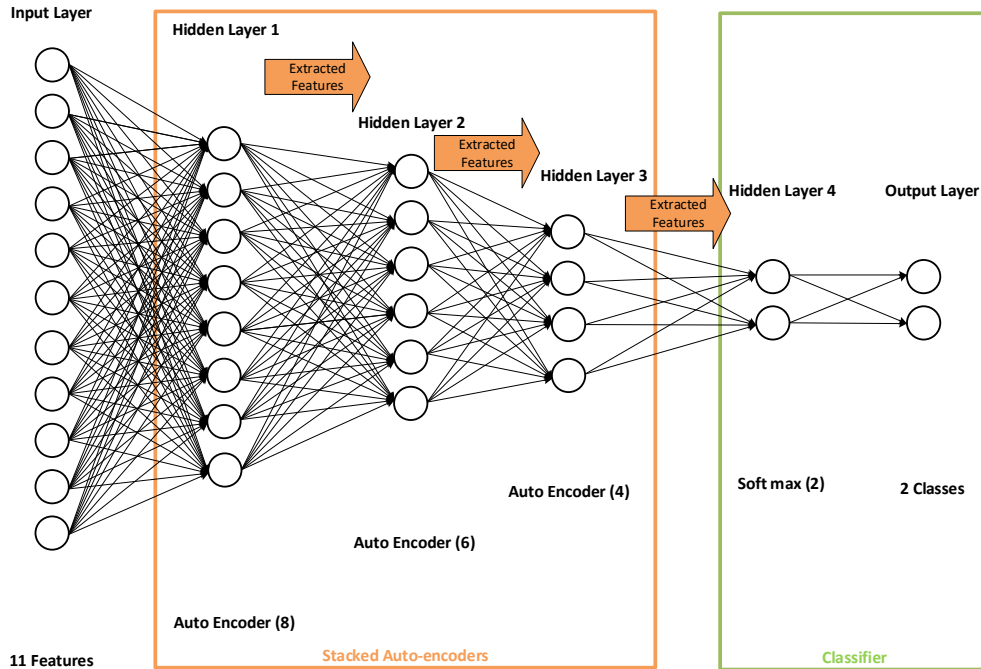


Figure 50: Implementation of SAE on injection attack dataset

The features extracted during the pre-training phase are purely unsupervised learning. The extracted features are given as an input to the SoftMax layer, which is the hidden layer 4. The SoftMax layer will use the 4 features and identify the normal and attack. The SMR layer is trained in a supervised manner with the extracted features and labels termed as fine-tuning step 1. Now, the whole SAE with all hidden layers (1-4) is fine-tuned via backpropagation in supervised manner. This procedure improves the extracted features for the learning of the SAE. This process is considered as fine-tuning step 2. After the process of fine-tuning, the trained network is validated with validation dataset. The validation of the training is performed on the entire network. Validation prevents the network from overfitting. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.8. Injection attack - Deep Belief Networks

The deep learning model of implementing the DBN on extracted feature dataset is represented in Fig. 51. From Fig. 51, we can see that the input layer with 11 nodes which represent the extracted features of the raw sensor data. The hidden layers 1, 2, 3 are used for complex feature extraction. During pre-training hidden layer 1 reduces the input features from 11 to 8 followed by hidden layer

2 which reduces the 8 features to 6 Features followed by hidden layer 3 that reduces the 6 features further to 4 features. Due to this reason, hidden layer 1 have 8 neurons, hidden layer 2 have 6 neurons and hidden layer 3 have 4 neurons. The output of the third hidden layers are considered as extracted features.

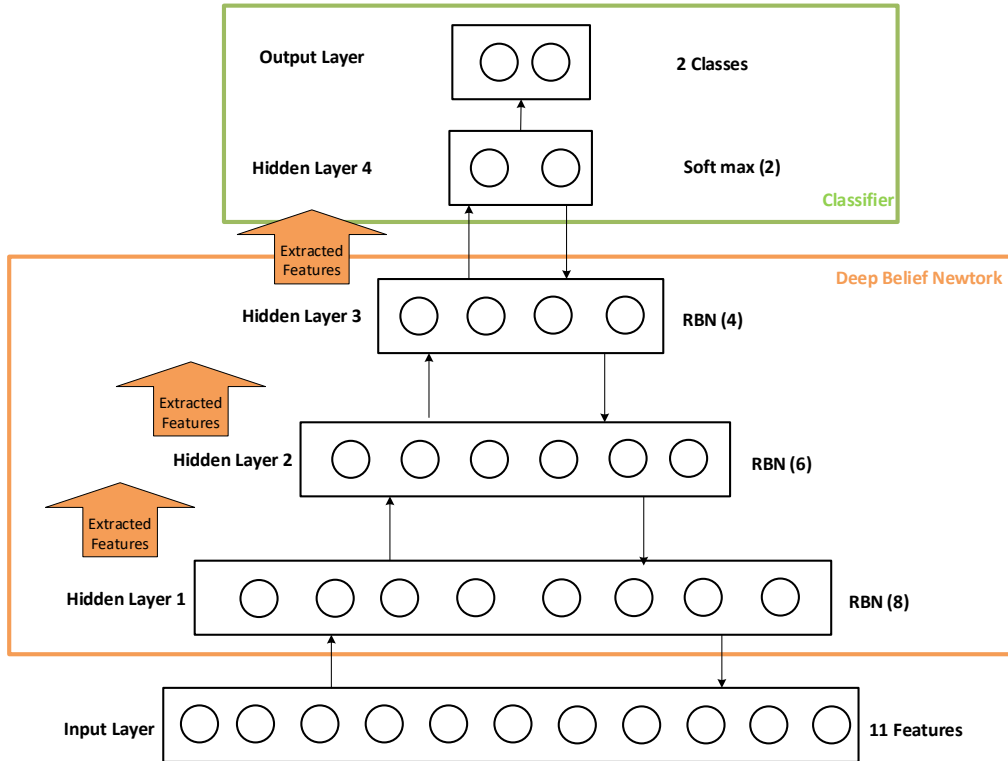


Figure 51: Implementation of DBN on injection attack dataset

The features extracted during the pre-training phase are purely unsupervised learning. The extracted features are given as an input to the SMR layer, which is the hidden layer 4. The SoftMax layer will use the 4 features and identify the normal and attack. The SoftMax layer is trained in a supervised manner with the extracted features and labels termed as fine-tuning step 1. Now, the whole DBN with all hidden layers (1-4) is fine-tuned via backpropagation in a supervised manner. This procedure improves the extracted features for learning of the DBN. This process is considered as fine-tuning step 2. After the process of fine-tuning, the trained network is validated with validation dataset. The validation of the training is performed on the entire network. Validation prevents the network from overfitting. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.6.9. Injection attack - Convolutional Neural Networks

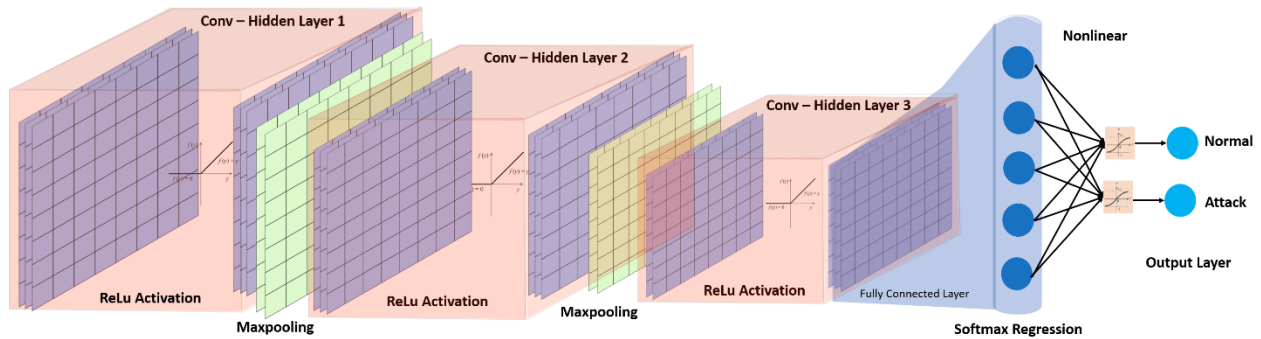


Figure 52: Implementation of CNN on injection attack dataset

From Fig. 52, we can see the implemented CNN model. It includes the following steps:

Step 1: The image data set of injection attack dataset obtained from the image representation stage with each image of size 8x8 is given as an input image to CNN – Hidden Layer 1.

Step 2: The CNN layers are initially initialized with random weights and filters and these are adopted during the training process.

Step 3: The network takes the input image and initiates the training process. The image goes through the forward propagation steps (convolution, ReLU and pooling operations along with forward propagation of the fully connected layers) and finds the output probabilities.

Step 4: The error value of the desired output to the generated output is calculated. And validation is performed after every 300 iterations.

Step5: Now, backpropagation with gradient decent is used to update the network weights and all filter values to minimize the output error.

The above steps are continued until the validation function measures the same value for five times as the patience was set to 5. This ensures the network from overfitting. Narrow convolution technique is used for the first convolution hidden layer. The output feature map of the first convolution hidden layer is smaller than 8x8. Due to this reason, in second and third hidden layers, wide convolution technique is used by padding the feature maps with zeros. SoftMax regression with non-linear sigmoid transfer function is used for classification of attack classes at the final fully

connected layer. The SMR layer is trained in a supervised manner with the extracted features and labels. The output of the CNN is either normal or an attack. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

5.7. Hybrid deep learning models

Every deep learning techniques has its own constraints during the extraction of features from the input data. Similarly, every classification algorithm such as SoftMax may not yield proper results for certain applications and development of anomaly detection based IDS is not an exemption. Some existing works on hybrid machine learning and deep learning models are already discussed during the literature review in Chapter 3. Due to this reason, hybrid deep learning models are proposed and implemented as a part of this thesis. The combinations of deep learning models verified in the scope of this thesis are listed in Table 5-5.

The implementation of hybrid deep learning models is similar to other deep learning implementations, but the key difference is the internal hidden layers used for feature extraction and the classification layer combination coupled to the deep learning model. The training of the hybrid models is performed according to the principles of validation and evaluated the validation accuracy. Later the best model is tested with the test data for evaluating the performance of the network with the performance metrics mentioned in Section 3.5. The results are discussed in Chapter 6 and in Appendix B.

Table 5-5: Different combinations of hybrid deep learning

Type No.	Feature Extraction Technique	Classification Technique
1	Stacked autoencoder	Support Vector Machines
2	Deep Belief Network	Support Vector Machines
3	Stacked Autoencoder + Deep Belief Network	SoftMax regression
4	Stacked Autoencoder + Deep Belief Network	Support Vector Machines

5.8. High performance models

Deep learning algorithms are computationally intensive. Due to the increasing number of data and the number of neurons in a deep learning algorithm makes it necessary to perform a lot of computations in a short period of time. Especially during the training process, the deep learning algorithms need to exploit the complex correlations between the input data and represent them as features. During the development of IDS using deep learning algorithms, it is found that the training takes a lot of time on normal CPU's. With the availability of modern high-performance CPU's and GPU's with higher clock speeds, it is possible to perform the parallel operations on these multi-core platforms.

The accelerated computing platforms are used especially helpful during the training phase. The pre-training, Fine-tuning step1 and fine-tuning step 2 are performed on multi-core high performance computing platforms with higher clock speeds and the performance is evaluated on different available platforms with different algorithms. The results are discussed in Chapter 6 and in Appendix B.

5.9. Attack Classification

Classification of attacks is directly obtained by supervised learning techniques as well as deep learning techniques. Most of the existing research classify the output of deep learning algorithm into two categories, namely normal and attack. In this thesis, classification of attacks into multiple attack classes is considered as it is necessary to know the attack type in order to take a preventive measure after attack identification. In this scope of this thesis, network attacks are detected with the different attacks such as DoS, Probe, R2L and U2R and injection attacks are classified to measurement injections and command injections.

5.9.1. NSL-KDD dataset

The proposed deep learning algorithms with necessary configurations are verified on NSL-KDD dataset. As the motivation of this thesis is classification of multiple attack classes, the training is also classified into different levels of attack identification based on the attack occurrence. The attack class classification is mentioned in the following Table 5-6.

From Table 5-6 in 2 class classification all the attack types were considered into single attack class apart from normal data and are represented as an attack. In 3 class classification DoS was considered separately and the rest of the attack types were considered into single attack class. In 4 class classification DoS and Probe were separated into two attack classes and R2L and U2R are considered as attacks. Finally, in 5 classes classification all the attack types were considered individually and represented by their attack names. The aim is to maximize the accuracy of detecting all attack classes but due to unbalanced data from Table 4-7, evaluation is also performed on the basis mentioned above.

Table 5-6: Attack class classification NSL-KDD dataset

Classes	Attack Types			
	DoS	Probe	R2L	U2R
2 Classes	X	X	X	X
3 Classes	-	X	X	X
4 Classes	-	-	X	X
5 Classes	-	-	-	-

5.9.2. UNSW-NB-15 dataset

The proposed deep learning algorithms with necessary configurations are verified on UNSW-NB-15 dataset. As the motivation of this thesis is classification of multiple attack classes, the training is also classified into different levels of attack identification based on the attack occurrence. The attack class classification is mentioned in the following Table 5-7.

From Table 5-7 in 2 class classification all the attack types were considered into single attack class apart from normal data and are represented as an attack. In 3 class classification, the next most prominent class, according to Fig. 18. Generic is considered separately and the rest of the attack classes are considered into single attack class. In 4 class classification, the next most prominent class, according to Fig. 18. Generic and Exploits are considered separately and the rest of the attack classes are considered into single attack class. This has continued further and finally, in 10 classes classification all the attack types were considered individually and represented by their attack names. The aim is to maximize the accuracy of detecting all attack classes but due to unbalanced data from Table 4-9, evaluation is also performed on the basis of considering non-prominent attack classes into single attack class. The flow of implementing SAE and DBN on UNSW-NB-15 dataset is a way similar to the implementation of SAE and DBN on NSL-KDD dataset. The only difference

is from UNSW-NB-15 dataset, 43 features were considered where are in NSL-KDD dataset we have 41 features.

Table 5-7: Attack class classification UNSW-NB-15 dataset

No. of Classes	Attack Type								
	Generic	Expl oits	Fuzz ers	DoS	Reconn aissance	Analysi s	Backd oor	Shell code	Worms
2 Classes	X	X	X	X	X	X	X	X	X
3 Classes	–	X	X	X	X	X	X	X	X
4 Classes	–	–	X	X	X	X	X	X	X
5 Classes	–	–	–	X	X	X	X	X	X
6 Classes	–	–	–	–	X	X	X	X	X
7 Classes	–	–	–	–	–	X	X	X	X

8 Classes	-	-	-	-	-	-	X	X	X
9 Classes	-	-	-	-	-	-	-	X	X
10 Classes	-	-	-	-	-	-	-	-	X

5.9.3. Injection attack dataset

For injection attack dataset, the attack classification is simple. As the output of the deep learning algorithm is binary in case of injection attack dataset, the output can be simply classified into either normal data or to a specific attack data i.e. normal or measurement injection attack or else normal or command injection attack.

6. Results and evaluation

This section presents the important results obtained through the implementation of the proposed approach using deep learning to secure ICS against cyber-attacks. The results of the individual deep learning algorithms on different datasets are discussed individually and presented in Appendix B by computing the necessary performance metrics. An evaluation is performed to identify the best algorithms for detection of attacks, followed by comparing the obtained results with the existing literature.

6.1. Selected results

This section provides a detailed evaluation based on the deep learning algorithms implemented in the scope of this thesis for securing ICS against network and injection attacks. The evaluation is differentiated into two section based on the types of attacks identified using deep learning models.

6.1.1. Deep learning models for network attacks

The deep learning algorithms performed efficiently in identifying the network attacks from the datasets. But each proposed algorithm has different detection accuracies for different attack types in different datasets.

NSL-KDD

Table 6-1: Classification accuracies of deep learning algorithms on NSL-KDD dataset

Type of Deep Learning Algorithm	Attack Classes					Over all Detection Accuracy (%)
	Normal (%)	DoS (%)	Probe (%)	R2L (%)	U2R (%)	
CNN	98.10	86.60	87.70	0	0	80.33
SAE	89.65	97.80	78.60	46.43	53.84	90.95
DBN	89.18	96.06	83.81	83.91	15.38	91.14

The summary of the proposed deep learning algorithms performance with best detection accuracies for different attack classes in NSL-KDD dataset is given in Table 6-1. Despite having the good detection accuracies for individual classes, the overall detection accuracy is low, this is due to the irregular distribution of attack samples in the dataset.

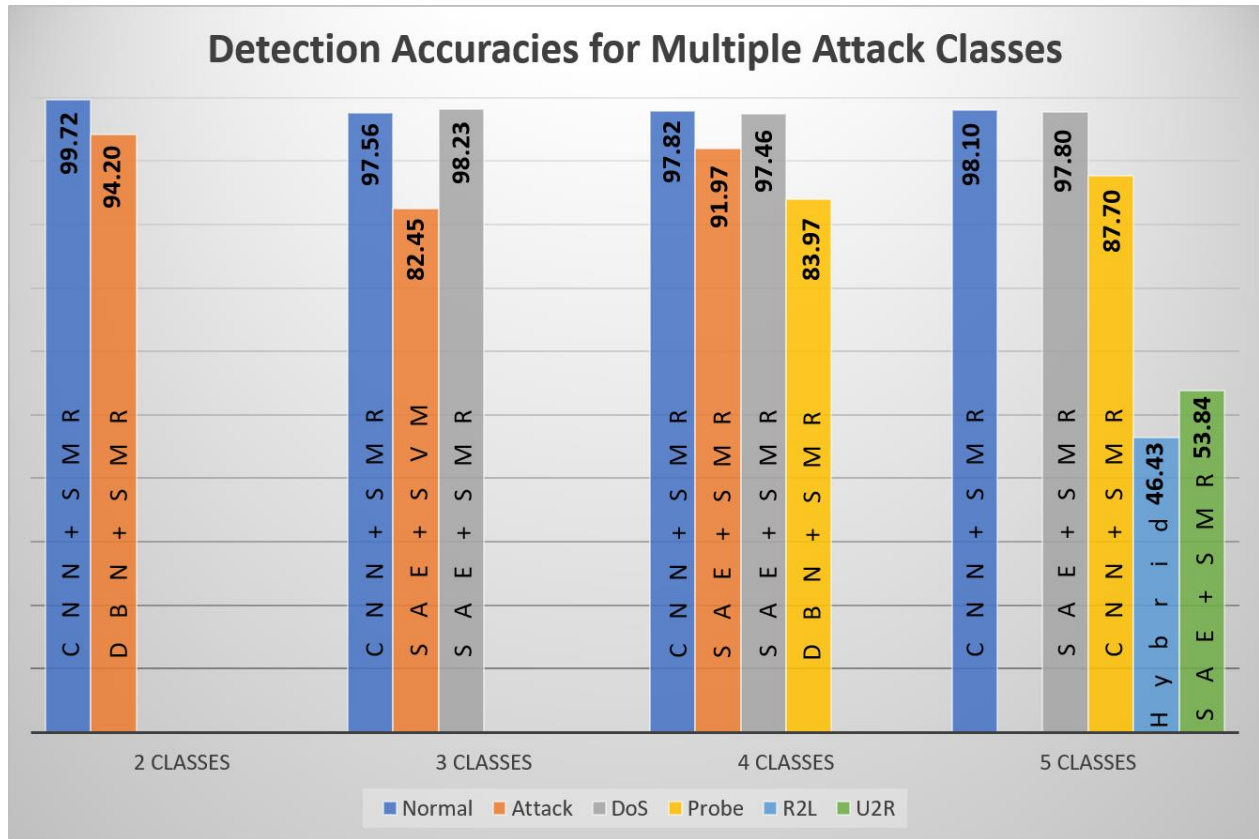


Figure 53: Overall detection accuracies of deep learning on NSL-KDD dataset

From Fig. 53, it is clear that the CNN algorithm was able to perform well for the detection of normal classes with higher accuracy in all categories. In the next level, SAE algorithms are good at detecting attack classes such as DoS, U2R attack classes. Detection of the attack class probe is not bad in comparison to CNN and DBN but DBN outperformed SAE and CNN for detection of probes in 4 class attack classes, but CNN performed well for probe attack in 5-class attack classification. The combination of multiple deep learning algorithm is found only advantageous in identifying the R2L attack class with accuracy, but its performance is limited in all other variations. The combination of SVM for deep learning seems advantageous if we look at 3-class attack

classification for identification general attack class. Due to this reason, the analysis on UNSW-NB-15 was only performed on the deep learning algorithms combined with the SMR.

UNSW-NB-15

The summary of the proposed deep learning algorithms performance with best detection accuracies for different attack classes in UNSW-NB-15 is given below. From Fig. 54, it is clear that the CNN algorithm performs well for a smaller number of classes with good accuracy. With the increase in classes, decrease in number of samples for other classes, the performance of the CNN is degraded. On the other hand, SAE performs good for multiple attack classes with better accuracy. Until 7 class classification, i.e., identifying, normal, generic exploits, fuzzers, DoS and reconnaissance SAE was able to detect them. From 8 class classification no algorithm performed well but SAE was able to detect some samples of analysis, backdoor and shellcode. Worms are the only attacks in the UNSW-NB-15 dataset, which were unable to be detected by and deep learning algorithm. It is obvious as the number of samples for attack class worms is too less in comparison to other classes. In some cases, DBN produced similar results to the SAE. But with an increase in attack classes, DBN was unable to maintain the accuracy for the attack classes with more samples. But the precision of DBN detection is higher in relation to SAE for the attack classes with more samples.

Benchmark

In order to evaluate the efficiency of the proposed approach with other existing approaches, a comparison of the obtained results is done with the existing techniques discussed in Chapter 3. As NSLK-KDD dataset is the most common dataset used for evaluating the performance of the algorithm, a benchmarking is also done using the NSL-KDD dataset. Most of the existing techniques evaluated the performance of the algorithm based on the over-detection accuracy of the algorithm by considering only 2-classes. Hence the benchmarking is also done using the same approach. The following Fig. 55. provides the overall detection accuracies of the existing techniques and our approach.

The red line in the Fig. 55 gives a comparison of the detection accuracy of the proposed deep learning approach with the other existing approaches. There exist some techniques which provide better detection accuracies than our approach, which crosses the red line. This is due to the use of

a single attack class for classification or using the same training dataset for training and testing of the machine learning technique.

Results and evaluation

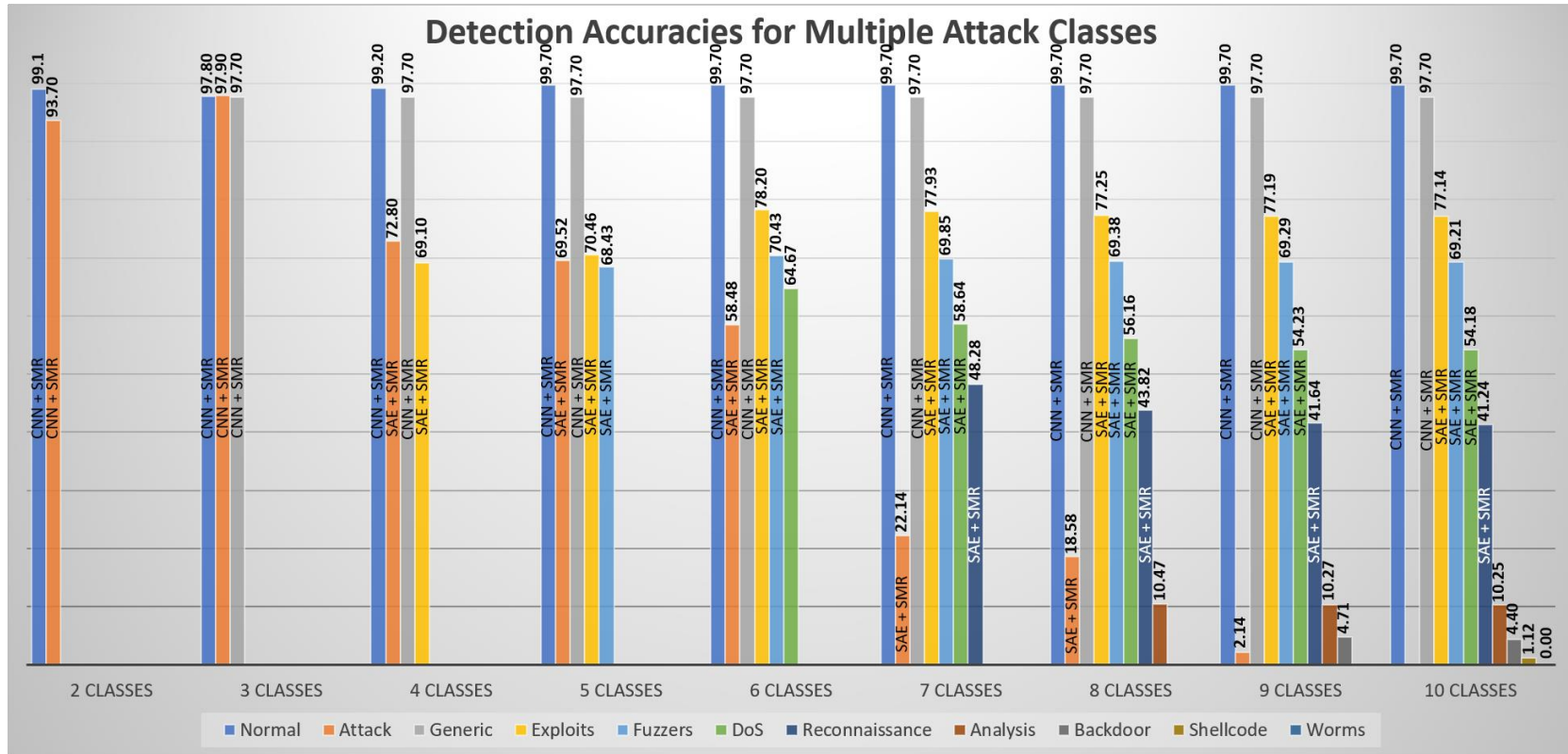


Figure 54: Overall detection accuracies of deep learning on UNSW-NB 15 dataset

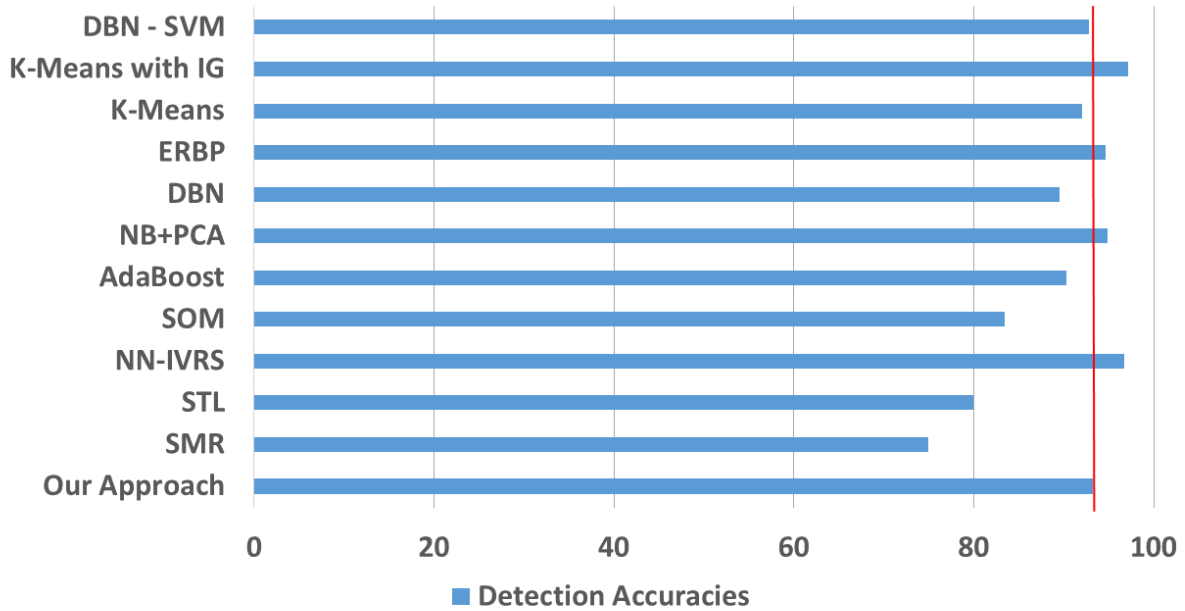


Figure 55: Benchmarking detection accuracies of NSL-KDD dataset with existing approaches

The proposed CNN architecture was able to perform much better than the existing CNN models. A benchmarking is performed with other existing techniques and the results look promising. The comparison is provided in Table 6-2.

Table 6-2: Benchmarking detection accuracies of proposed CNN algorithm on NSL-KDD dataset.

Technique	Accuracy (%)
CNN - ResNet 50	79.14
CNN - GoogLeNet	77.04
CNN - Proposed Approach	91.14

The benchmarking of the performance of deep learning was unable to perform on UNSW-NB dataset as limited research is performed using this dataset is available and due to the prominence of the NSL-KDD dataset, the latest research's still use the same for evaluating their algorithms rather than UNSW-NB-15. Due to the development of application specific CNN algorithm, our proposed algorithms were able to perform even better on UNSW-NB15 dataset compared to NSL-KDD dataset.

6.1.2. Deep learning models for injection attacks

The deep learning algorithms performed efficiently in identifying the injection attacks from the extracted basic features of the generated datasets. But each proposed algorithm has different detection accuracies for different attack types in different datasets.

Measurement injection

The summary of the proposed deep learning algorithms performance with best detection accuracies for identifying measurement injection attacks classes in generated process control plant dataset is given in Table 6-3.

Table 6-3: Classification accuracies of deep learning algorithms on measurement injection attacks

Type of Deep Learning Algorithm	Attack Classes		Over all Detection Accuracy (%)
	Normal (%)	Attack (%)	
CNN	97.50	94.90	96.70
SAE	97.44	96.69	97.18
DBN	86.39	88.51	87.11

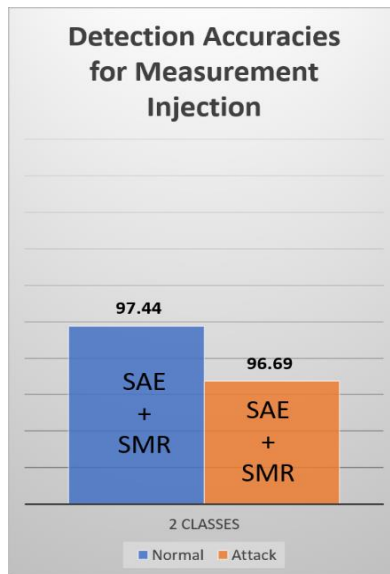


Figure 56: Overall detection accuracies of deep learning algorithms on measurement injection

From Fig. 56. we can see that, the performance of SAE is good for the detection of measurement injection attacks in relation to DBN and CNN. The detection accuracies of the CNN algorithm is nearly equal to the performance of SAE but DBN performs significantly low for the detection of measurement injection attacks.

Command injection

The summary of the proposed deep learning algorithms performance with best detection accuracies for identifying command injection attacks classes in generated process control plant dataset is given in Table 6-4.

From Fig. 57 we can see that, the performance of DNB is good for detecting normal data samples in command injection dataset while SAE performs better for identifying the command injection attacks. Even, the detection accuracy of the SAE for normal is almost equivalent to the detection accuracy of the DBN. DBN performance significantly poor for detection of the attack classes for command injection. The performance of CNN is comparable to the performance of SAE in identifying the command injection attacks.

Table 6-4: Classification accuracies of deep learning algorithms on command injection attacks

Type of Deep Learning Algorithm	Attack Classes		Over all Detection Accuracy (%)
	Normal (%)	Attack (%)	
CNN	95.90	92.70	94.90
SAE	97.07	94.11	96.09
DBN	97.45	57.05	84.13

Similar to UNSW-NB15 dataset, a benchmarking the performance of the injection attack dataset is not possible as the dataset had been generated for this specific application internally from the department. Nevertheless, the results from the proposed approach looks promising and implementation of deep learning algorithms in this domain brings a significant advantage.

Apart from mentioning the accuracies, additional performance metrics precision, recall, and F-measure of the individual algorithms were also analyzed. As detailed analysis of individual algorithms and their performance metrics are mentioned in Appendix B.

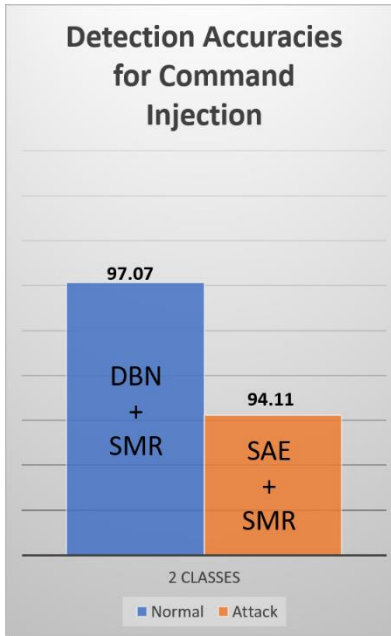


Figure 57: Overall detection accuracies of deep learning algorithms on command injection

6.1.3. Hybrid deep learning algorithms

The implementation of hybrid deep learning algorithms was performed on NSL-KDD dataset. A summary of the proposed hybrid algorithms performance with best detection accuracies for different attack classes in NSL-KDD dataset is given in Table 6-5.

Table 6-5: : Classification accuracies of different hybrid deep learning algorithms on NSL-KDD dataset

Type of Hybrid Deep Learning Algorithm	Attack Classes					Over all Detection Accuracy (%)
	Normal (%)	DoS (%)	Probe (%)	R2L (%)	U2R (%)	
SAE+SVM	91.47	96.05	78.60	65.72	19.23	91.71
DBN+SVM	92.06	95.33	75.19	35.57	46.15	91.23
SAE+DBN+SMR	90.56	95.79	78.21	49.74	21.11	90.92
SAE+DBN+SVM	89.90	95.36	81.03	86.43	44.23	91.02

6.1.4. High performance model

The proposed deep learning models are trained using the multi-core CPU's and GPU's for training the algorithm faster. The observations are mentioned in the Table 6-6 and Table 6-7.

Table 6-6: Training and Fine-tunings times of different deep learning algorithms on different CPU's

Technique	Intel 2 Quad Q 8400		Intel i7 – 4720 HQ	
	Pre-training (min)	Fine-Tuning (min)	Pre-training (min)	Fine-Tuning (min)
SA+SMR	157.25	7.36	123.73	5.94
SA+SVM	157.6	8.01	123.6	6.54
DBN+SMR	149.01	7.15	113.66	5.29
DBN+SVM	144.33	8.89	113.58	6.91

Table 6-6 gives an overview of the training time of different deep learning algorithms on two CPU's i.e., Intel 2 Quad Q 8400 and Intel i7 4720 HQ. Despite both being quad core processors, the number of threads on Q 8400 are 4 whereas on i7 4720 HQ there are 8 threads. This makes it a little faster in the training process. Both processors have a processor base frequency of 2.60 GHz. But i7 4720 HQ is equipped with max turbo frequency of 3.60 GHz. Q 4800 has 4MB cache where are i7 4720 HQ has 6MB cache. These are some parameters which make the performance improvements in the training process.

Table 6-7: Training of deep learning algorithms on different hardware acceleration platforms in serial and parallel modes

Hardware	Training Time (in Sec)	
Intel Core i7 – 4720 HQ	Serial	471.761888
	Parallel	147.423094
Intel Core i7 - 4790	Serial	260.049666
	Parallel	108.583120
Nvidia GeForce GTX 960M	132.5080829	
Intel Core i7 – 4720 HQ + Nvidia GeForce GTX 960M	145.593437	

Table 6-7 gives an overview and comparison of implementing the training process of a deep learning algorithms on CPU and GPU. As well as implementing the algorithms in a serial and parallel mode on the CPU's. With 8 threads, 3.60 GHz base frequency and 4.00 GHz turbo frequency and 8 MB cache makes the CPU i7 - 4790 much efficient in parallel mode of operation. This makes clear that the base frequency and the dataset types play a crucial role in the use of high-performance models.

6.2. Evaluation

All the algorithms discussed in the scope of this thesis can be interpreted as a complex form of the neural networks. The main difference between them would be the level of network hardwiring between the layers and on the training algorithms which they rely for updating the weights. In SAE, the information flow unidirectionally from the input layers, through the hidden layers, up to the output layers. Where as in DBN, the information flows both ways between the visible (input / output and the hidden layers) and the hidden layers. CNN differs from these two in different way. In CNN, instead of learning single global weight matrix between two layers, the aim to find a set of locally connected neuron. SAE and DBN are usable in applications with having huge sets of numerical data, whereas CNN are applicable on images or the datasets that are converted to the format of a 2D matrix, as is it done in the scope of this thesis.

SAEs are, in general good for the neural transformation to reconstruct the inputs in an efficient manner. The hidden layers learn the set of latent features. While DBN learns the joint probability of the input data using stochastic distribution. CNN learns the local correlation between the input data based on the filter kernels.

Based on the observations for the development of IDS using deep learning, it is clear that SAE and CNN outperform the DBN. This is due to the input data and network architecture. The given input samples are either features of a single network packet in case of network attacks or a set of features from 10 samples which already hold a correlation between the data samples. Due to this reason, DBN was unable to perform well in the learning process in comparison to SAE and CNN. SAE and CNN were able to extract the complex relations between the input samples and try to adjust the network based on the feedbacks.

It is important to understand that none of the deep learning algorithms by themselves are classifiers. They are coupled with either SMR or SVM. Despite the proved classification capabilities of SVM, SMR performed better in coupling with deep learning algorithms. Due to this reason, SMR are widely used in relation with deep learning in comparison to SVM.

The learning time of SAE algorithms is less in comparison to CNN. Therefore, the applications where a fast learning curve is necessary, SAE are better recommended in relation to CNN. Due to

the computing intensive convolution operations and a huge set of images, CNN consumers a lot of time and resources.

SAE are good and applicable for dataset with less amount of data but with more classes. From the results of SAE performance on UNSW-NB15 dataset, it is clear that SAE is capable of learning the attack classes even with less number of samples in contrast, CNN requires huge dataset but they learn more efficiently.

From the experience gained in the scope of this thesis some recommendations are made which correlated the input factors to the network infrastructures. The recommendations are scaled between 1 (Low) to 10 (High). Table 6-8 provides an overview of such recommendations.

Table 6-8: Recommendation of the deep learning algorithm in relation to the input parameters

Parameters / Network Type	SAE	DBN	CNN
No. of feature - More	7	5	10
No. of features - Less	9	6	5
No. of Outputs / Classes - More	7	6	9
No. of Outputs / Classes - Less	9	6	9
Types of features - More	7	5	9
Types of features - Less	9	6	6
Size of the dataset - More	8	7	10
Size of the dataset - Less	9	6	4
Network Hardwiring	6	7	10

From the above Table, we can see the DBN are not so much suitable for the development of cyber security strategies.

Along with the input parameters, the types of input parameters given to the deep learning algorithms also play a significant role in the detection accuracy. Table 6-9 provides the recommendation of using the specific deep learning algorithm for a specific input data type.

Table 6-9: Recommendation of the deep learning algorithm in relation to the input data types

Data types / Network Type	SAE	DBN	CNN
Numeric	8	7	10
Binary	3	3	8
Discrete	7	6	9
Continuous	8	7	10
Categorical	4	3	6
String	4	3	9

From the above recommendations, we can provide a way or an approach to build a deep learning-based security concept. For example, below Fig. 58 gives an approach in implementing the security of an ICS for injection attack.

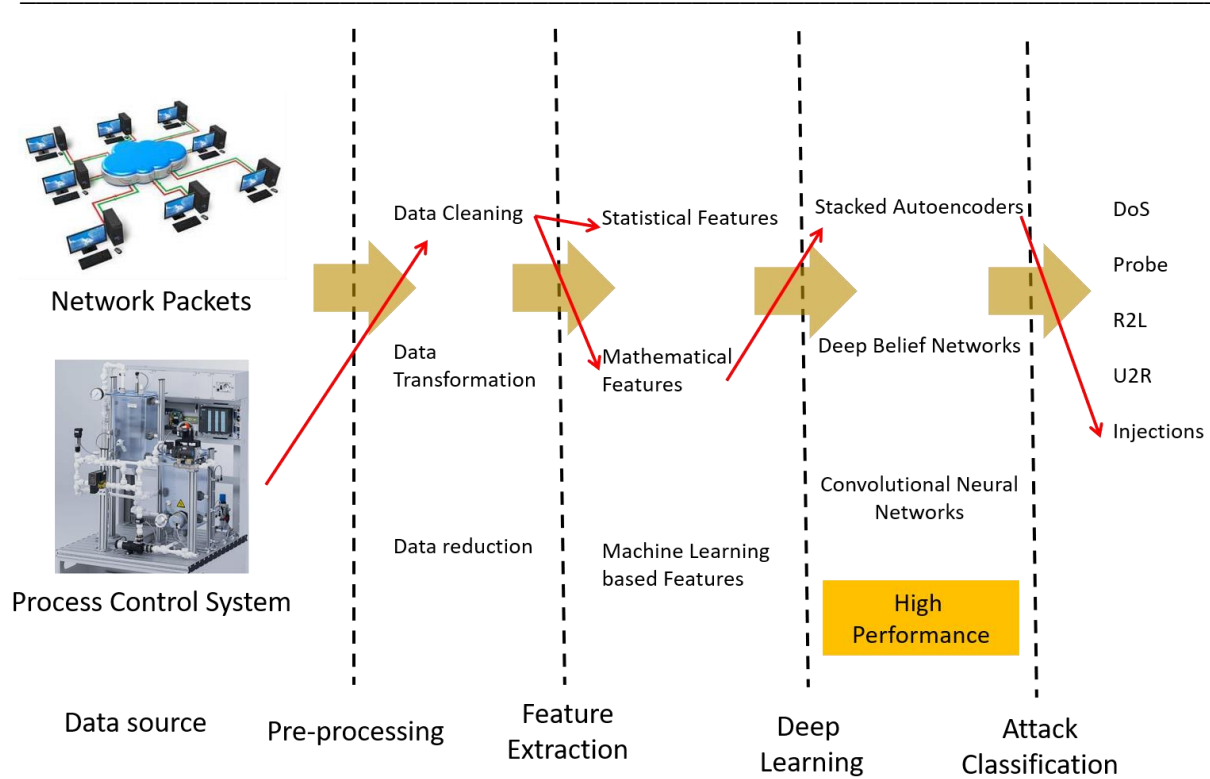


Figure 58: Sample recommendation of deep learning security concept on injection attacks

In Fig. 58, we can see the red arrows. These red arrows show the way of developing the security concept for injection attacks. From Fig 58, we can see that if we get the sensors data from a process control plant. We use the data cleaning approaches, such as filters and pre-process the data. After pre-processing, different statistical and mathematical features can be extracted from the sensor data. Based on the recommendations mentioned in Table 6-8 and 6-9, SAE can made as a choice if deep learning algorithms. The algorithm is trained with the extracted features and the classification in performed. As the dataset is not huge, the use of high-performance computing is not recommended for the injection attacks. Similarly, based on the properties of the input dataset, such recommendations can be achieved.

Regardless of many efforts in generalizing, every deep learning is application specific and data driven. The performance of the algorithm may vary based on the requirements and the data used. The above recommendations can be taken into consideration while developing an application based on deep learning algorithms.

7. False data injection attacks toolbox in MATLAB/Simulink

This section provides the efforts made in the direction and development of the injection attack toolbox simulated in MATLAB/Simulink. Proposed toolbox is available for download in GitHub [184]. The types of attack simulated are in accordance to Table 2-1.

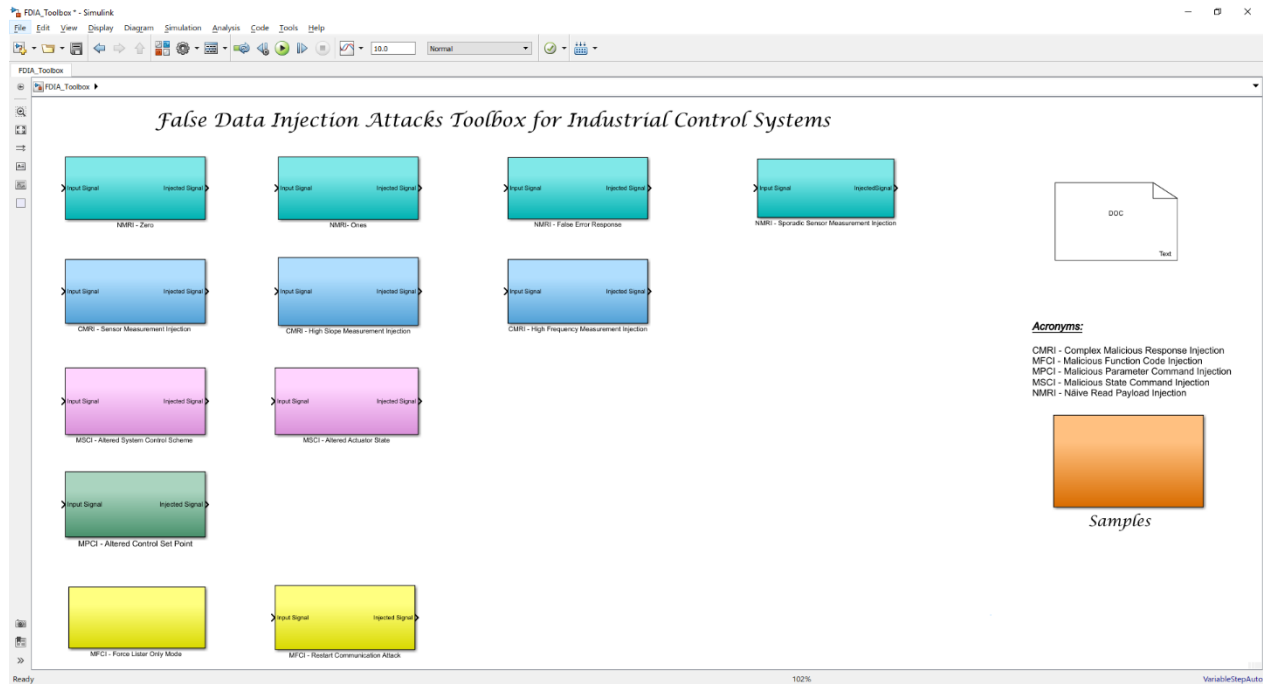


Figure 59: MATLAB/Simulink window of attack injection toolbox

7.1. Naïve Malicious Response Injection (NMRI) attacks

"Industrial control systems use polling techniques for continuously monitor the remote process. Polling is the process in which the transmission of queries is done between the client and server with a response packet from the server to the client. It provides the information about the Human Machine Interface (HMI) i.e. monitor and store the processed measurements and implement the response with reference to the processed measurements of the control parameters as it's the part of the feedback control loop [20]" [26].

"Here the complete process takes place with digital processing, which can be easily analysed by the systems. While in NMRI attack the systems for transmitting the data onto client to server and server to client which induced with process measurements along with invalid process measurements which effect the complete feedback control parameters" [26].

"The first injection attack can be originated from malicious control of PLC or RTU, while the second response injection attack can capture the network packets and alter the content of transmission of the server to client. The third response injection may be crafted and injected into the network by third party devices. Sometimes the response there might be multiple reposes to the client query about invalid response might be assumed due to a race condition or secondary attack such as denial of service attack which stops the server from responding" [26].

"An NMRI attack is defined as response injection attacks the sensor information and induce invalid sensor information, process measurement and also effects the feedback control loop process state. Dependent on the type of attack the NMRI attacks are classified in to four different types" [26].

Naïve Read Payload Size

"The Naïve Read Payload size attack is fully based on network protocol technology. This attack comes under the classification of NMRI attacks. The query about transactions in between the server and client, to send the number of objects or quantity fields should be returned exact same length of objects to it. Here, the exact data is unable to read due to a lack of information or details about each data object in the packets. Therefore, the few quantities of data or objects are returned as all zeros or all ones. Whenever the data cannot access the exact information about the objects, it is replaced by either zeros or ones. Fig. 60 provides the simulation results of the attack in relation to normal behaviour" [26].

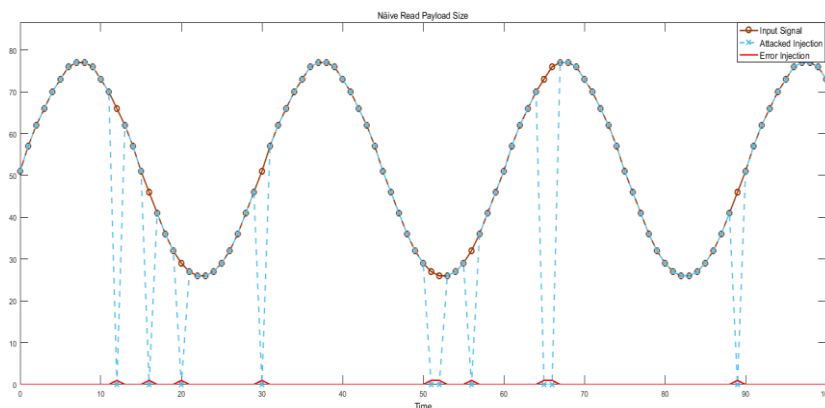


Figure 60: Naive read payload size attack behavior

Invalid Read Payload Size

"Here, the number of objects or data from the input register query is neglected. The response to the payload size is either larger quantity or smaller quantity of data objected than the requested number of objects of input. Whenever the input is not responded to the queries about the client side. The response to payload size may be decreased or extending the data size or by creating zeros, ones, or any random data into the valid payload size. In Fig. 61 the response of data objects which is neglected by the input query and which is returned into the all ones. This is shown as dotted lines from sine wave to value of data is ones. Cross notation describes about the error response in the size of the valid payload attack" [26].

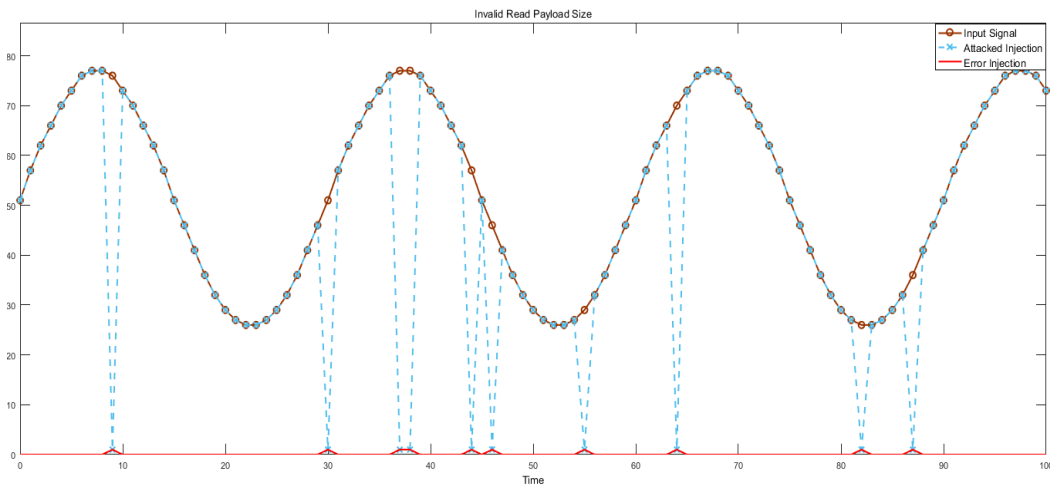


Figure 61: Invalid read payload size attack behavior

Naive False Error Response

"The term itself defines, the sender sends the packets to the client and the packets are returned to the client after read command which is a falsified error value. The NMRI attack can send the random codes to the client, which fall into the legal range codes. Otherwise, it comes under the out of range of given random codes. Simply, it is outside of the legal range values" [26].

"From Fig. 62, the injected response is denoted by the cross-dotted lines. When the attack occurs, the random codes are either in legal range or outside of the range. Here, legal range is from 20 to 70 and data codes due to error response in the system the codes are returned into below or legal range" [26].

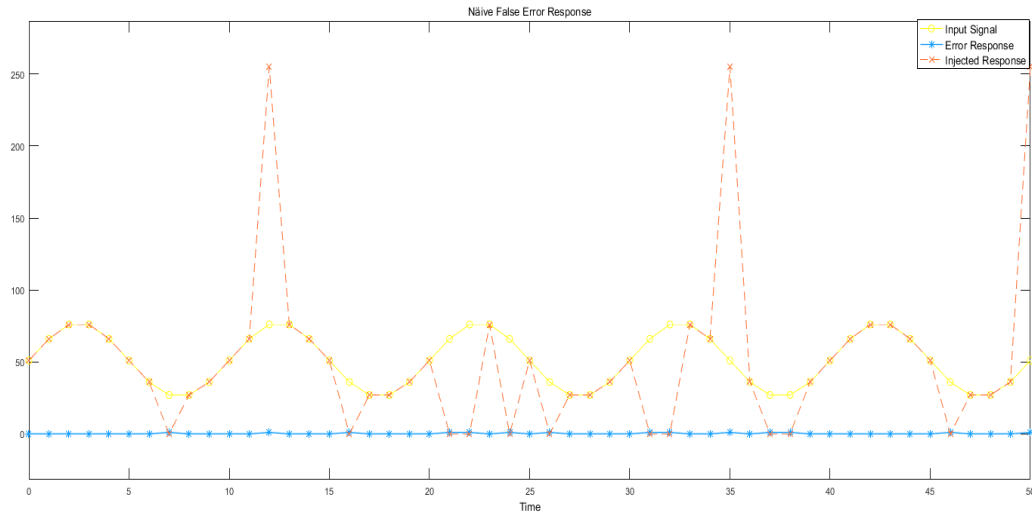


Figure 62: Naïve false error response attack behavior

Sporadic Sensor Measurement Injection

"It is an NMRI attack which sends the sporadic false process measurement injection, which resides outside the bounds of High (H) and Low (L) control set points but doesn't raise up to an alarm set point bounds of High High (HH) and Low Low (LL). Generally, the motor on and off cycle are decided by the High (H) and Low (L) set points of the controller, which are received from the sensor. There is a physical delay while switching between on and off of the motor which are outside the setpoints of High (H) and Low (L). But, the NMRI attack differs from the rest as response injects sporadic process measurements that affect the tank's setpoints" [26].

"From Fig. 63, the input signal is the motor signal and the attacked injections are clearly depicted. The injection system defines the sporadic process measurements, which affect the sensor to detect the changes in the set points High (H) and Low (L). The shift in the water level from 12 to 8 to different set points in the graph at specific error points represents the injection of sporadic process measurement, which intern changes the tank's water level" [26].

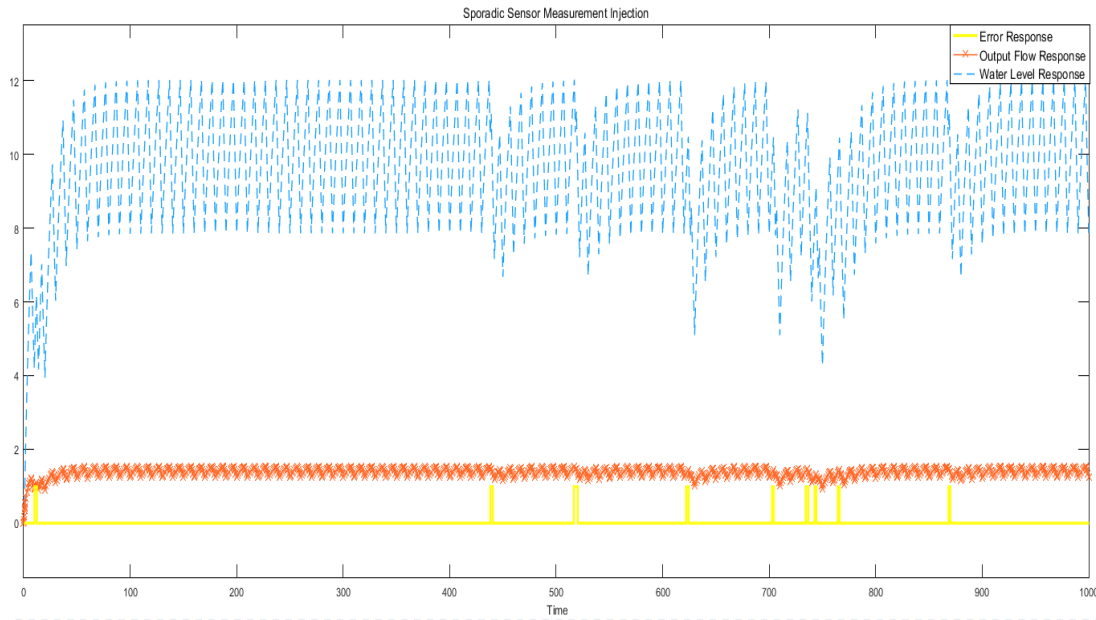


Figure 63: Sporadic sensor measurement attack behavior

7.2. Complex Malicious Response Injection (CMRI) attacks

"These attacks a bit highly sophisticated than NMRI and require a greater understanding of the CPS. They always tend to mask the physical response, which affects the negative feedback control loop [20]" [26].

Slope Sensor Measurement Injection

"It is a CMRI attack in which the calculated process measurements are injected. The sensor measurement might increase or decrease the trend based on the system. The attack masks the actual state of the physical process by masking the feedback of the control loop. The attacker sends the pre-calculated process measurements that make the system shift from normal behaviour to abnormal behaviour and tend to shift the system in critical state" [26].

"The graph is taken from the example of the water tank system. From Fig. 64, the sensor gets effected, which intern affects the physical change such as water level. The physical change of 12 to 5 sets a different setpoint. This effect will also change the behaviour of the motor switching cycle. If the sensor sends the water flowed lower than the normal High (H) the motor turns off and when it reaches to lower Low (L) set point the motor should turn on. This switching behaviour will affect the set points and replaces with new set points. The plot represents the change in behaviour

of the motor switching cycle with changes to the behaviour of the motor at attacks injected and the shift in motor switching is clearly visible" [26].

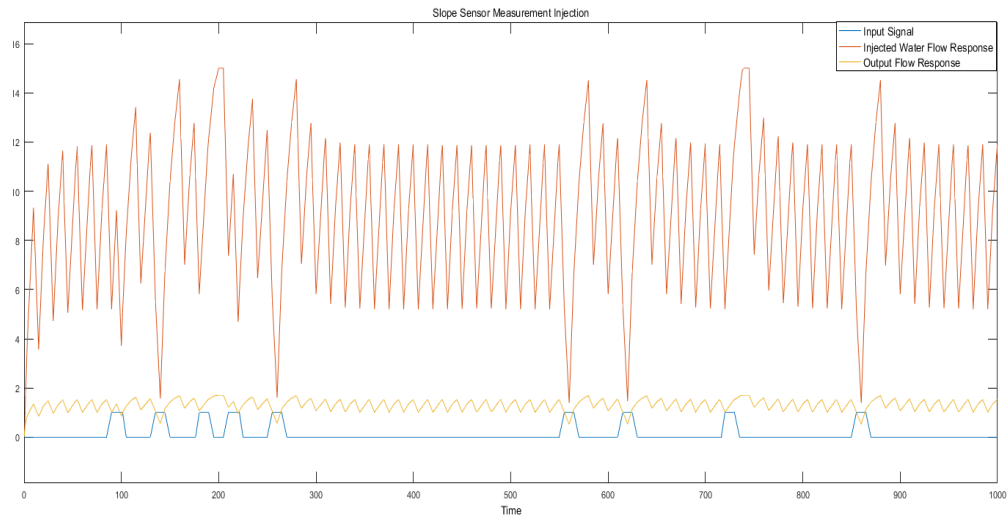


Figure 64: Slope sensor measurement attack behavior

High slope measurement injection

"It is a CMRI attack that sends repeatedly process measurements containing the same measurements to mask the real state of the system. The process measurement parameters are completely captured and then the same signal is repeated to make the impression that the system is running normally to the client" [26].

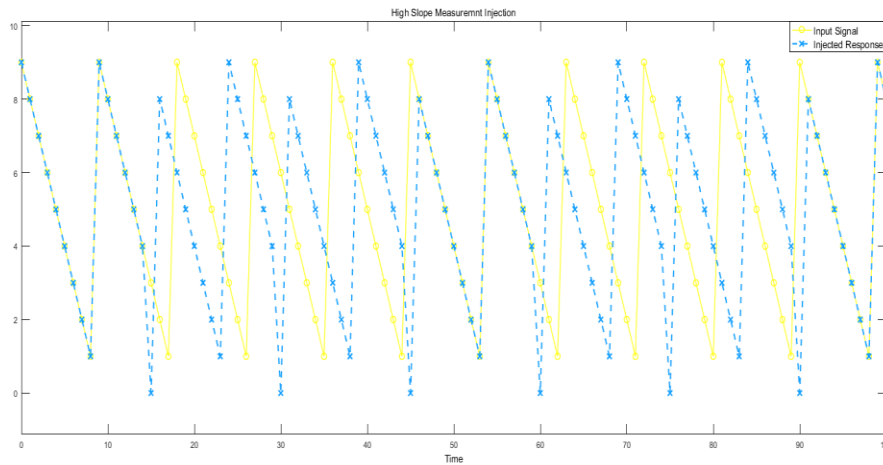


Figure 65: High slope measurement attack behavior

"The simulating of the attack is primarily focussed on the capturing of the sensor signal and replaying it. From Fig. 65 we can see that the signal is captured for an instance of time and then the same signal is repeated. The representation of the original signals is represented in crosses and the repeated signal is represented as circles" [26].

High Frequency Measurement Injection Attack

"This CMRI attacks attempt to mask the original system physical behaviour, which is controlled negatively to affect the control loop system and it is managed by cyber physical behaviour system. In this attack, the measurement process of frequency will increase to the extended values than the normal rate. Frequency attacks look like normal system functionality. But it changes the system behaviour, which changes the process due to masking" [26].

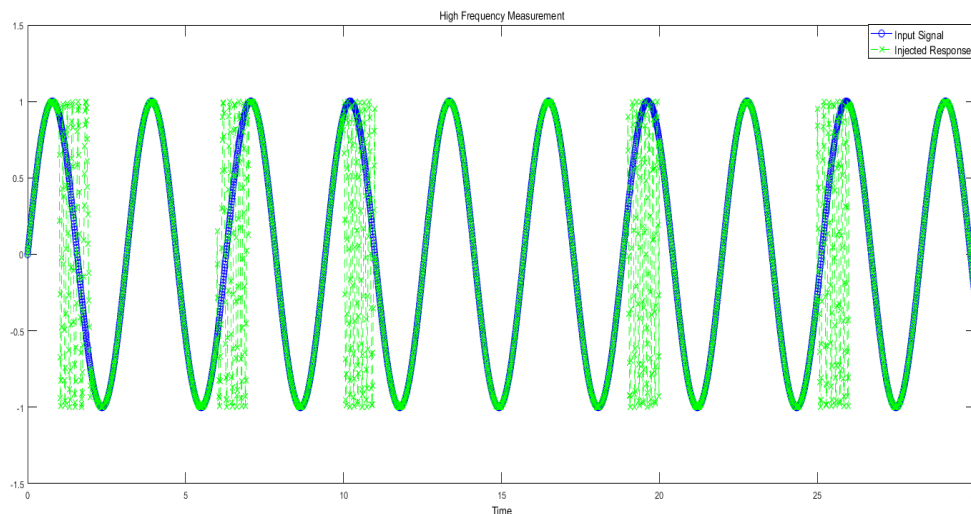


Figure 66: High frequency measurement injection attack behavior

"Fig. 66. for example, consider the water tank storage system. The graph shows the measurements are changing in the water storage tank level before and during the frequency injection attack. The injected response (cross-dotted line) is represented during water tank system under attack and the normal input signal is without being attacked. Whenever the frequency is injected suddenly, the water liquid level is increasing and decreases rapidly. Because this attacked system appears to the normal system behaviour in a different time, and also handle the misconfigured system by an operator. Such a simulated period of water level changes during high demand which in reality, the system is being attacked due to the falsified response" [26].

7.3. Malicious State Command Injection (MSCI) attacks

"These attacks inject invalid commands which cause incorrect control actions in the feedback control loop. These attacks mostly occur where human intervention is minimum as supervisory control takes over all the integral part and operated by few operations limiting the human interface. In these systems, hackers have a greater advantage of injecting the false supervisory control parameters in the control system network. Most of the remote terminals and intelligent electronic devices are generally programmed to monitor the system automatically and control the physical process when needed or via the remote interface. So, the hackers include a similar logic and take control of the control system, and these attacks are termed as Command Injection attacks" [26].

"The effects of the command injection attacks are

- Send false control and false configuration commands to alter the system behaviour.
- It impacts the loss of process control and also interrupts the device communication.
- It affects the unauthorized modifications of device configurations.
- It also includes unauthorized modifications of process set points.

Command injections are grouped into three categories as Malicious State Command Injection (MSCI), Malicious Parameter Command Injection (MPCI), and Malicious Function Code Injection (MFCI)" [26].

Altered System Control Scheme

"It is an MSCI attack which changes the control mode from automatic to manual. In automatic mode, the complete system is operated on PLC control, but in manual mode, each and every operation should be monitored and controlled by the operator. It leads to various parameter changes which affect the system response. The increase or decrease in any parameter could trigger an alarm response which also includes the operator to take control. After switching into the manual mode, the operator loses the full control of the system and able to make changes w.r.t. the control parameters" [26].

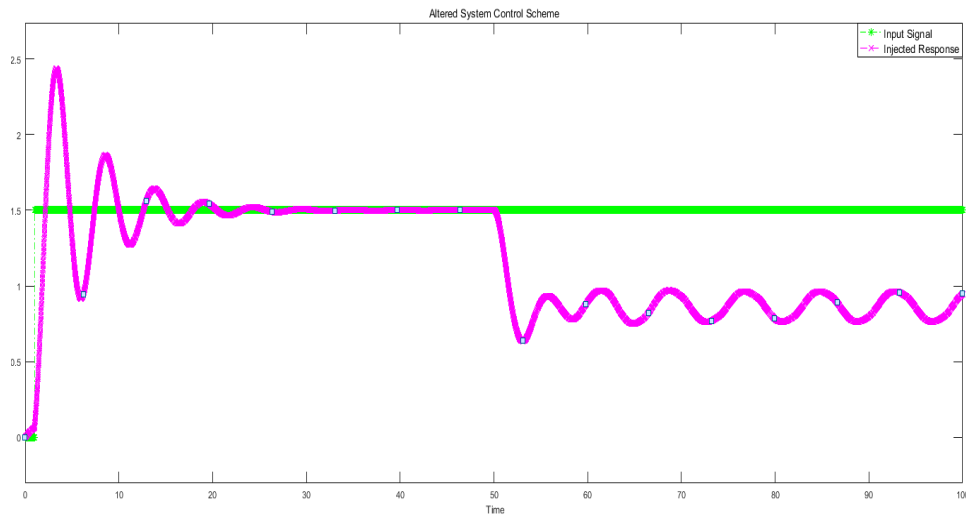


Figure 67: Altered system control scheme behaviour

"From Fig. 67, the system is represented as a signal of PID controller obtaining the saturation value. After saturation, it shifts from automatic to manual. The manual mode is represented by a different signal" [26].

Altered Actuator State

"It is an MSCI attack. The change of actuator state from on to off or vice versa. The change of actuator is done continuously. The change of actuator will also affect the system behaviour and make changes into the physical devices. It is represented in Fig. 68" [26].

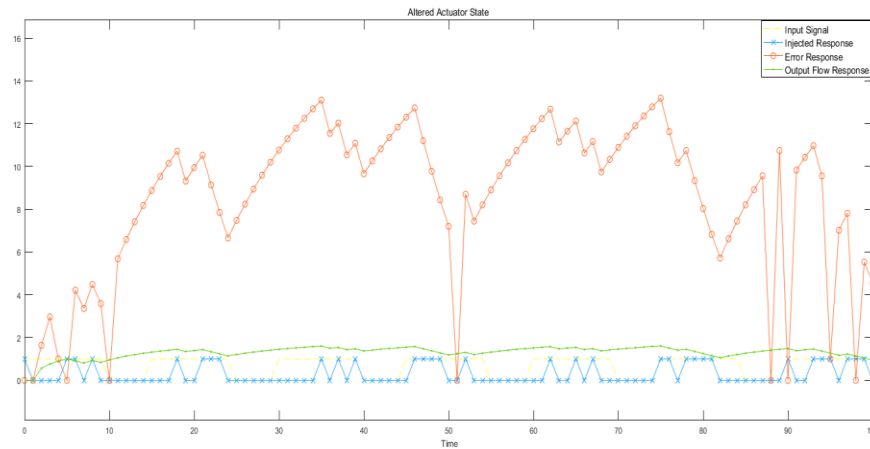


Figure 68: Altered actuator state behaviour

7.4. Malicious Parameter Command Injection (MPCI) attacks

"Here the attack parameters are changed and it causes the system to perform incorrect control actions. Here the setpoints are changed which leads to other changes in the system behaviour" [26].

"The tank model described above is considered, and modifications are done w.r.t the attack. In the attack, the state of the actuator must change from ON to OFF or OFF to ON along with the physical change associated to it. Fig. 69, describes the implementation of the tank system actuator state. The change in the actuator is represented by crossed line (injected response) while the original actuator state is represented by the dotted line (Input signal)" [26].

Altered Control Set Points

"It is an MPCI attack. The changes in the setpoints of the tank in the control schema of the feedback control system occur. The change in the set points lead to the change in the system behaviour and may trigger other critical response signals. The operator doesn't have any idea of changes in the system but the values of the system itself. The change in system behaviour may damage the system or leads to work the plant in the critical state which could lead to sudden failure of the complete plant" [26].

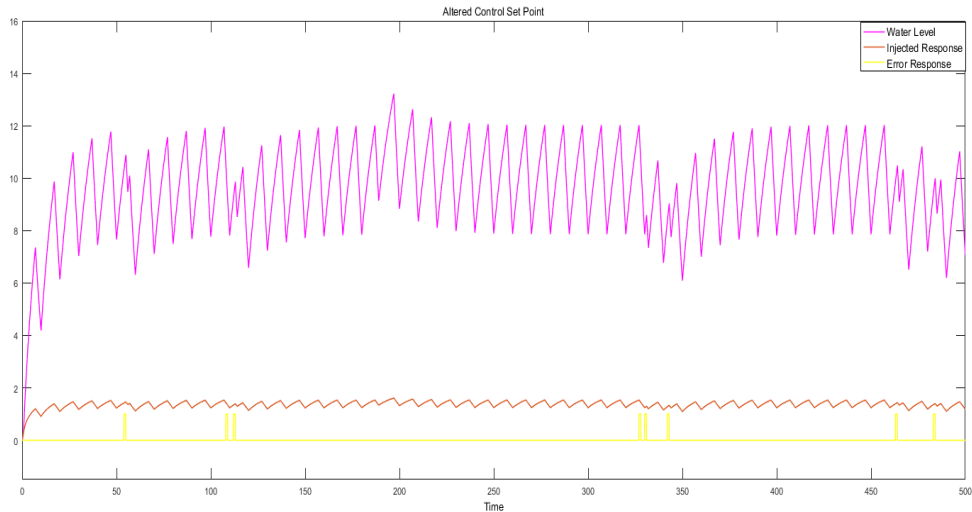


Figure 69: Altered control set points behaviour

"Fig. 69. represents the changes in the tank set points takes place which is represented in straight line. The change in set points High (H) and Low (L) in the water tank. The change occurs at instances where the injection response takes places. The other physical changes such as a change in output are also represented in the figure with a thick line. The water tank model is preferred, and with the change of the water level set points, the physical change of the motor switching and the feedback loop is also affected" [26].

7.5. Malicious Function Code Injection (MFCI) attacks

Force Listen Only Mode

"This Malicious Function Code Injection (MFCI) attacks comes under the subclasses of command injection attack. This attack causes a server to no longer transmits the data to the network. Normally, most of the industrial control systems are using polling techniques, such as HMI. The term itself says, machine to machine communication. These interfaces normally well designed in the combination of hardware and software products and these are enabled to all users to provide inputs which are converted into machine language signals. Then, these signals are processed and provide the desired response to the users. HMI called with different names, namely computer-human interface, human-machine interface. That means, HMI software checks the status of the data periodically from servers. These interface software's display the data to the human operators.

So that, they can control the supervisory actions based upon the state of the current system. This attack is made through the MATLAB GUI" [26].

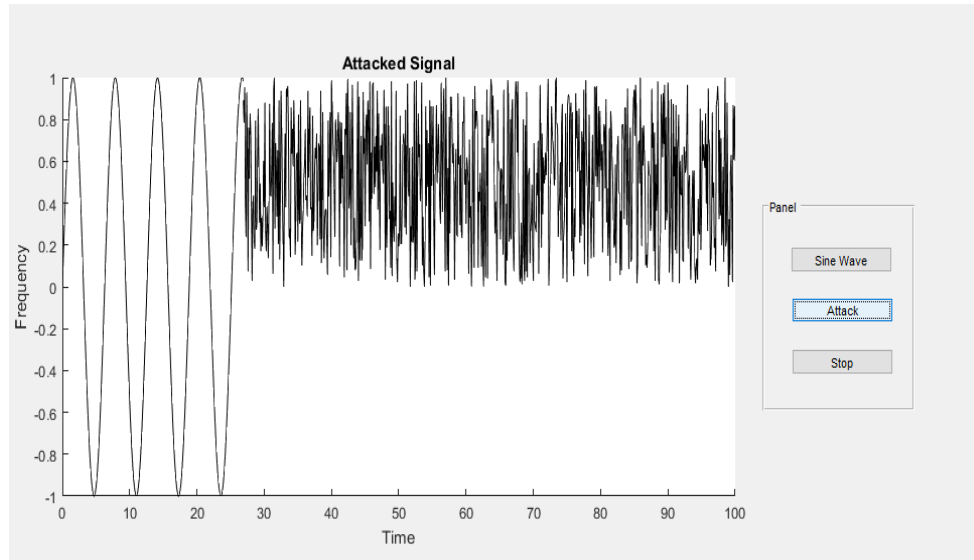


Figure 70: Force listen only mode behaviour with GUI

"The signal queries sent by the client to the server. The server accepts the data and sends the acknowledgement and response to the queries to the client. This kind of interaction of various devices which are designed in a way human to machines handling and vice-versa is done reliability. Whenever the system data is affected by the third party (attacker) in between the server and client. The attacker controls the queries response to the client from the server. This leads to loss of visibility of data and control. As shown in Fig. 70. the attacker induced some sort of data into the system's data. Therefore, the client unable to get the results of his queries and lose the data visibility and control of the data. That means the data is received by server, but it is read only mode due to the hands-on control by attacker" [26].

Restart Communication

"The restart communication defines as, the data is in communication between from sender to receiver. The data restarts irrespective of time before it receives the entire data completely to the receiver's side, is called restart communication. While the process of communication of data is unexpectedly restarted and it leads to temporary loss of data. Due to this, the data causes lags in

communication and during this time it provide the default value and no communication takes place in a meanwhile. These data lags lead to major interrupts in the communication in between them" [26].

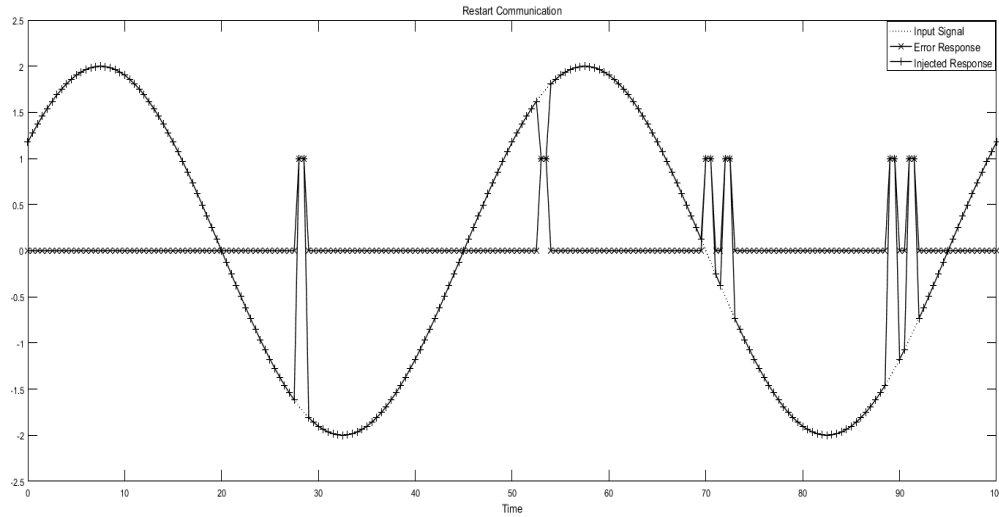


Figure 71: Restart communication attack behavior

"From Fig. 71, we can observe sine wave signal, which is modified from sender to receiver. During the modification of the signal, communication is restarted at multiple time intervals. The attacker injects the error communication, which is denoted by the straight line and error response is the crossed-straight line. In between the data is lost due to restarting the system which leads to loss of data and also effects the system communication" [26].

8. Conclusion and Future work

In this chapter, the summary of the research is detailed within the scope of this thesis. The work has demonstrated the use of applying the deep learning algorithms for securing the ICS against the cyber-attacks. For this purpose, the available and relevant literature is reviewed. Later, the proposed implementations were applied on the different datasets. The performance of the algorithms is evaluated in detail for network and injection attacks using different metrics. The results from the proposed approach looks promising. This chapter also presents some initial thoughts about the directions for future research.

8.1. Conclusion

Cybersecurity in ICS infrastructures gains in size and complexity at a fast pace. Their protection priorities exceed the existing threats processing capabilities. Hence, it is crucial to investigate methods which not only achieve high detection accuracy but are also capable in handling the current network traffic and data flow demands. To achieve these trending demands, the security strategies should bring intelligence, self-adaption and automatic novel threat detection.

The key issues of this thesis are:

- Exploring the existing research on the intrusion detection and multi-class classification
- Development of deep learning algorithms for network attack detection and multi-class attack classification
- Development of deep learning algorithms for injection attacks in ICS
- A toolbox for the simulation of multiple injection attacks dataset in ICS

From the available literature, this thesis broadly classifies the possible attacks on ICS into two categories, i.e. network attacks and injection attacks. These attacks are further classified into different attack types based on the attacks presented in the available datasets. A detailed description about these attacks and their effects on the security priorities of ICS are discussed in detail in Chapter 2.

As security on ICS is not a new area of research, it found that there exists a huge amount of research for the development of IDS using different ML techniques for the identification of network attacks. The most relevant literature is discussed in Chapter 3. From this, it is observed that firstly, either these researchers not considering the complete dataset while training or not considering the attack

classes for classification. Secondly, existing mechanisms have huge false positives even for classification of normal network behaviour. Another finding from the literature review is, there exist very seldom research work in the development of security strategies against injection attacks in ICS infrastructure.

This thesis has proposed an approach of extracting different complex features out of the network packets and datagrams to understand the behaviour of normal and attack behaviour in ICS. Different deep learning algorithms such as SAE, DBN and CNN were implemented for extraction of proper features. These algorithms are coupled with different classification algorithms like SoftMax regression and SVM. A discussion on the proposed concept and the usage of different deep learning algorithms is mentioned in Chapter 5. It discusses the basics of deep learning algorithms followed by the implementations with detailed architectural details.

Initially, the developed algorithms are trained with the training dataset and are validated. The validation is performed efficiently to avoid overfitting and underfitting of the network. These trained deep learning algorithms are tested with the test datasets. Different performance metrics such as accuracy, precision, Recall and F-Measure were computed and the efficiency of the algorithms is evaluated.

The developed algorithms are trained and tested with different datasets for evaluation. NSL-KDD and UNS-NB-15 datasets were used for the development of deep learning algorithms against network attacks. The details about the dataset and the attack types present in this dataset are discussed in Chapter 4. For the development of deep learning algorithms against injection attacks, a dataset is simulated using a process control plant. Injection attacks such as command injection and measurement injection were incorporated into the dataset efficiently. A more detailed discussion on the dataset is done in Chapter 4.

The summary of obtained results were discussed in Chapter 6. A detailed analysis of different deep learning algorithms for intrusion detection on different datasets is performed. The analysis shows that the deep learning algorithms are efficient in detecting the attacks more accurately in comparison to the existing machine learning models discussed in the literature. From the evaluation, it is clear that each algorithm has its efficiency for certain types of attacks on certain

datasets. Hence, some recommendations are made in the evaluation section which algorithm can perform good on which types of input datasets and their characteristics.

After the development of the deep learning algorithms for securing ICS against injection attacks, it is observed that in order for further development of these security mechanisms it is necessary that the proposed deep learning algorithms need to train for more attack classes. For this purpose, an injection attack toolbox is developed and discussed in Chapter 7. This toolbox paves the way for future developers to generate the dataset based on their application and requirements.

Although the proposed deep learning algorithms are efficient in identifying the attacks efficiently in comparison to the existing machine learning algorithms, to match with the existing network and data bandwidths, it is still necessary to work on the false positives. Handling of an imbalanced dataset through the extraction of proper features is done until some extent. But, after observing the results it is found that more effort in the direction of handling imbalanced dataset is necessary for more efficient intrusion detection.

Finally, we can conclude that the intrusion detection in ICS is a highly active research area. Deep learning algorithms are helpful and efficient in fulfilling the task of intrusion detection and play a major role in securing ICS from novel attacks. During this thesis, some efforts are made in this direction, but still some research questions and challenges do exist.

8.2. Future work

The work performed in this thesis provides a basis for future research of developing the IDS in ICS domain. One of the future works is applying a broad range of features for anomaly-based intrusion detection. These features need to be flow-based and calculated in real-time to enable the detector to handle the existing gigabit network speeds. Furthermore, a customized deep learning algorithm should be devised to minimize the CPU and memory consumptions of the intrusion detectors. It is also advantageous to work on an efficient random sampling method to reduce the huge number of flows that are fed to the system as the training dataset.

One of the areas that need a lot of improvement is the fusing different algorithms in developing an effective intrusion detection system. Techniques such as Dempster's rule of combination is proved to be good, assigning the probabilities are quite challenging. This can be developed by either applying more features or utilizing a more efficient clustering techniques.

As discussed in thesis, the training of an individual deep learning algorithms requires a lot of training time. Acceleration of the training process is an important requirement in present world due to the availability of new threat every day. Although the use of acceleration platforms such as GPU's improve the training speed, development of algorithm specific hardware for deploying the developed IDS in industrial networks is advantageous. Hence, the research can also be carried out in this direction.

Despite several unsupervised techniques exist for the development of IDS, this thesis concentrated mainly on the supervised approaches to decrease the false positives and get better accuracies. Use of efficient clustering algorithms by cascading multiple ML algorithms such as SOM with HMM or Fuzzy C Means Clustering with Decision trees can improve the detection of anomalies/intrusions that are not foreseen.

Most of the injection attacks detection algorithms are trained with limited datasets that are generated or simulated offline. In reality, ICS generated a huge amount of data per day. So, looking into the direction and application of handling big data strategies such as data storage, parallel pre-processing and efficient feature extraction techniques are necessary.

All of the proposed algorithms and methods in this thesis could only be used for a static dataset of IDS. None of them could be directly applied as a real-time intrusion detection system. Real-time IDS require a response in time and it usually deals with data according to the instance. Deep learning algorithms could be used on real-time IDS, but it needs more improvements and testing in practice.

References

- [1] claire-ai.org, “Quotes.” [Online]. Available: <https://claire-ai.org/quotes/>. [Accessed: 10-Jan-2021].
- [2] Eric. D. Knapp. Joel Thomas Langill, *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*, 2nd ed. Syngress, 2014.
- [3] Adi Dar, “Protecting Industrial Control Networks – It’s Not Just About SCADA Security,” *Cyberbit*, 2017. [Online]. Available: <https://www.cyberbit.com/blog/ot-security/protecting-industrial-control-networks-scada-security/>. [Accessed: 21-Aug-2019].
- [4] V. Paliath and P. Shakarian, “Modeling cyber-attacks on Industrial Control Systems,” in *IEEE Conference on Intelligence and Security Informatics (ISI)*, 2016.
- [5] Kaspersky, “FIVE MYTHS OF INDUSTRIAL Control Systems Security,” vol. 37, no. October, pp. 2012–2013, 2012.
- [6] N. Cuppens-bouahia and D. Hutchison, *Security of Industrial Control Systems and Cyber-Physical Systems*. 2016.
- [7] A. Colbert, Edward J. M., Kott, *Cyber-security of SCADA and Other Industrial Control Systems*, 63rd ed. Springer, 2016.
- [8] V. D. Qishi Wu, Sajjan Sjiva, Sankardas Roy, Charles Ellis, “On modeling and simulation of game theory-based defense mechanisms against DoS and DDoS attacks,” in *Proceedings of the 2010 spring simulation multiconference*, Society for Computer Simulation International, 2010, p. 159.
- [9] Threatcloud Intelligence, “Live Cyber Attack Threat Map,” *Check Point Software Technologies Inc.* [Online]. Available: <https://threatmap.checkpoint.com/ThreatPortal/livemap.html>. [Accessed: 21-Aug-2019].
- [10] R. Nigam, “(Known) SCADA Attacks Over the Year,” *Security Research*, 2015. [Online]. Available: <https://blog.fortinet.com/2015/02/12/known-scada-attacks-over-the-years>. [Accessed: 15-Sep-2017].
- [11] N. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., & Weaver, “The spread of the sapphire/slammer worm.”
- [12] Denis Legezo, “Operation Ghoul: Learning from the targeted attack analysis to protect your business,” 2016. [Online]. Available: <https://www.kaspersky.com/blog/ghoul/5897/>. [Accessed: 15-Sep-2017].
- [13] Mark Thompson, “Iranian Cyber Attack on New York Dam Shows Future of War,” 2016. [Online]. Available: <http://time.com/4270728/iran-cyber-attack-dam-fbi/>. [Accessed: 15-Sep-2017].
- [14] Omar Santos, *End-to-End Network Security: Defense-in-Depth*. Cisco Press, 2007.

References

- [15] Nouredine. Boudriga. Mohamed Hamdi, “Computer and network security risk management: Theory, challenges, and countermeasures,” *Int. J. Commun. Syst.*, vol. 18, no. 8, pp. 763–793, 2005.
- [16] Jarmo Mölsä, “Mitigating denial of service attacks: A tutorial,” *J. Comput. Secur.*, vol. 13, no. 6, pp. 807–837, 2005.
- [17] E. Cole, R. Krutz, and J. W. Conley, *Network security bible*, 2nd ed. John Wiley & Sons, 2011.
- [18] Manjusha. D. Priya U. Kadam, “Various Approaches for Intrusion Detection System: An Overview,” *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 2, no. 11, 2014.
- [19] Sandro. Etalle. Damiano Bolzoni, “Approaches in anomaly-based intrusion detection systems,” in *1st Benelux Workshop on Information and System Security*, 2006.
- [20] T. Morris and W. Gao, “Industrial Control System Cyber Attacks,” *Int. Symp. ICS SCADA Cyber Secur. Res.*, pp. 22–29, 2013.
- [21] J. F. Karen Scarfone, Keith Stouffer, *Guide to Industrial Control Systems (ICS) security: Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC)*. National Institute of Standards & Technology, 2011.
- [22] Helge. Janicke. Allan Cook, Richard SMith, Leandros Maglaras, “Measuring the risk of cyber attack in industrial control systems,” in *4th International Symposium for ICS & SCADA Cyber Security Research*, 2016.
- [23] Tomohisa. Hayakawa. Cetinkaya, Ahmet, Hideaki Ishii, “An overview on denial-of-service attacks in control systems: Attack models and security analyses,” *Entropy*, vol. 21, no. 2, p. 210, 2019.
- [24] Sasanka Potluri; Shamim Ahmed; Christian Diedrich, “Securing Industrial Control Systems from False Data Injection Attacks with Convolutional Neural Networks,” in *Development and Analysis of Deep Learning Architectures*, Springer, 2019, pp. 197–222.
- [25] Nikhil. S. Mangrulkar, “Network Attacks and Their Detection Mechanisms : A Review,” 2014, vol. 90, no. 9, pp. 36–39.
- [26] Sasanka Potluri, Christian Diedrich, Sai Ram Roy Nanduru, Kishore Vasamshetty “Development of Injection Attacks Toolbox in MATLAB/Simulink for Attacks Simulation in Industrial Control System Applications,” in *IEEE 17th International Conference on Industrial Informatics (INDIN)*, 2019.
- [27] Y. Mo and B. Sinopoli, “False data injection attacks in control systems,” *IEEE Conf. Decis. Control*, 2010.
- [28] Anurag K. Srivastava Ren Liu, Ceeman Vellaithurai, Saugata S. Biswas, Thoshitha T. Gamage, “Analyzing the cyber-physical impact of cyber events on the power grid,” *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2444–2453, 2015.
- [29] Frederic Lemieux, *Current and Emerging Trends in Cyber Operations: Policy, Strategy and*

-
- Practice*. Springer, 2015.
- [30] Sakir. S.ezer Rafiullah Khan, Peter Maynard, Kieran McLaughlin, David Laverty, “Threat Analysis of BlackEnergy Malware for Synchronphasor based Real-time Control and Monitoring in Smart Grid,” *ICS-CSR*, vol. 16, pp. 1–11, 2016.
- [31] David Hollis, “Cyberwar case study: Georgia 2008,” *Small Wars Foundation*, 2011.
- [32] D. McMillen, “Attacks Targeting Industrial Control Systems (ICS) Up 110 Percent,” *Security ntelligence*, 2016. [Online]. Available: <https://securityintelligence.com/attacks-targeting-industrial-control-systems-ics-up-110-percent/>. [Accessed: 16-Sep-2019].
- [33] CISA - Cyber Infrastructure, “Overview of Cyber Vulnerabilities,” *Industrial Control Systems*. [Online]. Available: <https://www.us-cert.gov/ics/content/overview-cyber-vulnerabilities>. [Accessed: 16-Sep-2019].
- [34] Robert. A. Martin, “Managing vulnerabilities in networked systems,” *Computer (Long Beach. Calif)*., vol. 34, no. 11, 2001.
- [35] Sajal Bhatia, Nishchal Kush, Chris Djamaludin, James Akande, and Ernest Foo, “Parctical Modbus Flooding Attack and Detection,” in *12th Australasian Information Security Conference, AISC*, 2014.
- [36] Jasna Markovic-Petrovic and Mira Stojanovic, “Analysis of SCADA system vulnerabilities to DDoS attacks,” in *11th International Conference onTelecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, 2013.
- [37] R. Anderson, “Network Attack and Defense,” in *Security Engineering*, Second., Wiley, 2008.
- [38] Giovanni Kemmerer and Richard A Vigna, “Intrusion detection: a brief history and overview,” *Computer (Long Beach. Calif)*., vol. 35, no. 4, pp. 127–130, 2002.
- [39] Sandeep Kumar, “Classification and detection of computer intrusions,” 1995.
- [40] Sanjay Sharma and RK Gupta, “Intrusion detection system: A review,” *Int. J. Secur. Its Appl.*, vol. 9, no. 5, pp. 69–76, 2015.
- [41] Alexey A Titorenko and Aleksey A Frolov, “Analysis of modern intrusion detection system,” in *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIconRus)*, 2018, pp. 142–143.
- [42] Batista Maria Carolina, Gustavo EAPA Prati, and Ronaldo C Monard, “A study of the behavior of several methods for balancing machine learning training data,” *ACM SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20–29, 2004.
- [43] Oualid Koucham, “Intrusion detection for industrial control systems,” 2018.
- [44] Ashfaq. Hussain. Farooqi *et al.*, “Intrusion Detection Systems for Wireless Sensor Networks: A Survey,” vol. 56, pp. 234–241, 2009.
- [45] Tripti Sharma and Sinha Khomlal, “Intrusion detection systems technology,” *Int. J. Eng. Adv. Technol.*, vol. 1, no. 2, pp. 28–33, 2011.

References

- [46] Patel A, Qassim Q, Shukor Z, and Nogueira J, “Autonomic agent-based self-managed intrusion detection and prevention system,” in *Proceedings of the South African Information Security Multi-Conference (SAISMC 2010)*, 2011, pp. 223–234.
- [47] Patel Ahmed, Taghavi Mona, Bakhtiyari Kaveh, and Junior Joaquim Celestino, “An intrusion detection and prevention system in cloud computing: A systematic review,” *Journal Netw. Comput. Appl.*, vol. 36, no. 1, pp. 25–41, 2013.
- [48] Debar Herve, Dacier Marc, and Wespi Andreas, “Towards a taxonomy of intrusion-detection systems,” *Comput. Networks*, vol. 31, no. 8, pp. 805–822, 1999.
- [49] Pharate Abhishek, Bhat Harsha, Shilimkar Vaibhav, and Mhetre Nalini, “Classification of Intrusion Detection System,” *Int. J. Comput. Appl.*, vol. 118, no. 7, 2015.
- [50] Karthik M, Pravin R. Patil, Ajay kaurav, S.Sibi Chakkaravarthy, R.Patil, Pravin, “Intrusion Detection system: A Review of the state of the art,” *IOSR J. Comput. Eng.*, vol. 16, no. 1, pp. 108–112, 2014.
- [51] Forrest Stephanie, Hofmeyr Steven A, Somayaji Anil, and Longstaff Thomas A, “A sense of self for unix processes,” in *IEEE Symposium on Security and Privacy*, 1996, pp. 120–128.
- [52] Anderson Debra, Frivold Thane, and Valdes Alfonso, “Next-generation intrusion detection expert system (NIDES): A summary,” *SRI Int. Comput. Sci. Lab. Menio Park*, 1995.
- [53] H. Kozushko, “Intrusion detection: Host-based and network-based intrusion detection systems,” vol. 11, 2003.
- [54] Lee Wenke, Stolfo Salvatore J, and Chan Philip K, “Learning patterns from unix process execution traces for intrusion detection,” in *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, 1997, pp. 50–56.
- [55] Sandeep Kumar and Eugene H Spafford, “A software architecture to support misuse intrusion detection,” 1995.
- [56] Y. Weinsberg, S. Tzur-David, D. Dolev, and T. Anker, “High performance string matching algorithm for a network intrusion prevention system (NIPS),” *2006 Work. High Perform. Switch. Routing*, p. 7 pp., 2006.
- [57] Varun Chandola, “Anomaly detection for symbolic sequences and time series data,” 2009.
- [58] Jyothsna K Munivara, VVRPV Prasad, and VV Rama Prasad, “A review of anomaly based intrusion detection systems,” *International J. Comput. Appl.*, vol. 28, no. 7, pp. 26–35, 2011.
- [59] Tom M Mitchell, *Machine Learning*, Engineerin. Mcgraw-hill Science, 1997.
- [60] Thomas Cover and Peter Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [61] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas, “Supervised machine learning: A review of classification techniques,” *Emerg. Artif. Intell. Appl. Comput. Eng.*, vol. 160, pp. 3–24,

-
- 2007.
- [62] Wenke Lee, Salvatore J Stolfo, and Kui W Mok, "A data mining framework for building intrusion detection models," in *IEEE Symposium on Security and Privacy*, 1999, pp. 120–132.
- [63] Jacek M Zurada, *Introduction to artificial neural systems*, 8th ed. West publishing company St. Paul, 1992.
- [64] Gang Wang, Jinxing Hao, Jian Ma, and Lihua Huang, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6225–6232, 2010.
- [65] Rex A Dwyer, "A faster divide-and-conquer algorithm for constructing Delaunay triangulations," *Algorithmica*, vol. 2, no. 1–4, pp. 137–151, 1987.
- [66] Emilio Corchado and Alvaro Herrero, "Neural visualization of network traffic data for intrusion detection," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2042–2056, 2011.
- [67] Yu Yao, Yang Wei, Fu-xiang Gao, and Ge Yu, "Anomaly intrusion detection approach using hybrid MLP/CNN neural network," in *6th International Conference on Emerging Technologies (ICET)*, 2006, pp. 1095–1102.
- [68] Thomas G Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, 2000, pp. 1–15.
- [69] Aikaterini Mitrokotsa and Christos Dimitrakakis, "Intrusion detection in MANET using classification algorithms: The effects of cost and model selection," *Ad hoc Networks*, vol. 11, no. 1, pp. 226–237, 2013.
- [70] Chris Sinclair, Lyn Pierce, and Sara Matzner, "An application of machine learning to network intrusion detection," in *15th Annual Computer Security Applications Conference (ACSAC'99)*, 1999, pp. 371–377.
- [71] Srilatha Chebrolu, Ajith Abraham, and Johnson P Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Comput. Secur.*, vol. 24, no. 4, pp. 295–307, 2005.
- [72] SHolom M Weiss and Nitin Indurkha, "Decision tree pruning: biased or optimal?," in *AAAI*, 1994, pp. 626–632.
- [73] Siva S Sivatha Sindhu, Suryakumar Geetha, and Arputharaj Kannan, "Decision tree based light weight intrusion detection using a wrapper approach," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, 2012.
- [74] Ron Kohavi and George H John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1–2, pp. 273–324, 1997.
- [75] Gary Stein, Bing Chen, Annie S Wu, and Kien A Hua, "Decision tree classifier for network intrusion detection with GA-based feature selection," in *43rd annual Southeast regional conference-Volume 2*, 2005, pp. 136–141.
-

References

- [76] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, “A practical guide to support vector classification,” *Taipei*, 2003.
- [77] Vladimir Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.
- [78] Bernhard Schölkopf, “The kernel trick for distances,” in *Advances in neural information processing systems*, 2001, pp. 301–307.
- [79] Yi Liu and Yuan F Zheng, “One-against-all multi-class SVM classification using reliability measures,” in *IEEE International Joint Conference on Neural Networks*, 2005, pp. 849–854.
- [80] Jason Weston and Chris Watkins, “Support vector machines for multi-class pattern recognition,” in *Esann*, 1999, pp. 219–224.
- [81] Yanxin Wang, Johnny Wong, and Andrew Miner, “Anomaly intrusion detection using one class SVM,” in *5th Annual IEEE SMC Information Assurance Workshop*, 2004, pp. 358–364.
- [82] Gisung Kim, Seungmin Lee, and Sehun Kim, “A novel hybrid intrusion detection method integrating anomaly detection with misuse detection,” *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1690–1700, 2014.
- [83] Xian Rao, Chun-Xi Dong, and Shao-Quan Yang, “An intrusion detection system based on support vector machine,” *J. Softw.*, vol. 4, no. 14, 2003.
- [84] Feng Luo, Latifur Khan, Farokh Bastani, I-Ling Yen, and Jizhong Zhou, “A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles,” *Bioinformatics*, vol. 20, no. 16, pp. 2605–2617, 2004.
- [85] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, and Tzong-Wann Kao, “A novel intrusion detection system based on hierarchical clustering and support vector machines,” *Expert Syst. Appl.*, vol. 38, no. 1, pp. 306–313, 2011.
- [86] Dong Seong Kim, Ha-Nam Nguyen, and Jong Sou Park, “Genetic algorithm to improve SVM based network intrusion detection system,” in *19th International Conference on Advanced Information Networking and Applications*, 2005, pp. 155–158.
- [87] Soo-Yeon Ji, Bong-Keun Jeong, Seonho Choi, and Dong Hyun Jeong, “A multi-level intrusion detection method for abnormal network behaviors,” *J. Netw. Comput. Appl.*, vol. 62, pp. 9–17, 2016.
- [88] Michael Gastpar, Pier Luigi Dragotti, and Martin Vetterli, “The distributed karhunen--loeve transform,” *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5177–5196, 2006.
- [89] Jaina Patel and Krunal Panchal, “Effective intrusion detection system using data mining technique,” *J. Emerg. Technol. Innov. Res.*, vol. 2, no. 6, pp. 1869–1878, 2015.
- [90] Adel Nadjaran Toosi and Mohsen Kahani, “A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers,” *Comput. Commun.*, vol. 30, no. 10, pp. 2201–2212, 2007.

References

- [91] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.
- [92] Anna L Buczak and Erhan Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.
- [93] Jayveer Singh and Manisha J Nene, "A survey on machine learning techniques for intrusion detection systems," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 11, pp. 4349–4355, 2013.
- [94] . Revathi S. Malathi, A, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, 2013.
- [95] Sasanka Potluri; Shamim Ahmed; Christian Diedrich, "Convolutional Neural Networks for Multi-class Intrusion Detection System," in *6th International Conference on Mining Intelligence and Knowledge Exploration, MIKE*, 2018.
- [96] Tuan. A. Tang, Lotfi. Mhamdi, Des. McLernon, Syed. Ali. Raza. Zaidi, and M. Ghogho, "Deep learning approach for Network Intrusion Detection in Software Defined Networking," *2016 Int. Conf. Wirel. Networks Mob. Commun.*, pp. 258–263, 2016.
- [97] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," *Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol. (formerly BIONETICS)*, 2016.
- [98] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1185–1190, 2012.
- [99] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, no. c, pp. 21954–21961, 2017.
- [100] Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion Detection using Deep Belief Networks," pp. 339–344, 2015.
- [101] Sasanka Potluri, Christian Diedrich "Deep Feature Extraction for Multi-class Intrusion Detection in Industrial Control Systems," *Int. J. Comput. Theroy Eng.*, vol. 9, no. 5, pp. 374–379, 2017.
- [102] Sasanka. Potluri and Christian. Diedrich, "Accelerated deep neural networks for enhanced Intrusion Detection System," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2016, vol. 2016-Novem.
- [103] Sasanka. Potluri, Navin. F. Henry, and Christian. Diedrich, "Evaluation of hybrid deep learning techniques for ensuring security in networked control systems," *2017 22nd IEEE Int. Conf. Emerg. Technol. Fact. Autom.*, pp. 1–8, 2017.
- [104] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018.

References

- [105] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion Detection Using Convolutional Neural Networks for Representation Learning,” pp. 858–866, 2017.
- [106] R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Applying convolutional neural network for network intrusion detection,” *2017 Int. Conf. Adv. Comput. Commun. Informatics*, pp. 1222–1228, 2017.
- [107] Yao Liu, Peng Ning, and Michael K Reiter, “False data injection attacks against state estimation in electric power grids,” *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, p. 13, 2011.
- [108] Yuan Chen, Soumya Kar, and Jose MF Moura, “Optimal attack strategies subject to detection constraints against cyber-physical systems,” *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1157–1168, 2017.
- [109] Ziyang Guo, Dawei Shi, Karl Henrik Johansson, and Ling Shi, “Optimal linear cyber-attack on remote state estimation,” *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 1, pp. 4–13, 2016.
- [110] A. S. Willsky, “A survey of design methods for failure detection in dynamic systems,” *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.
- [111] Zong. Han. Yu and Wen-Long. Chin, “Blind False Data Injection Attack Using PCA Approximation Method in Smart Grid,” *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1219–1226, 2015.
- [112] Kamesh and N. Sakthi Priya, “Security enhancement of authenticated RFID generation,” *Int. J. Appl. Eng. Res.*, vol. 9, no. 22, pp. 5968–5974, 2014.
- [113] Sasanka Potluri; Christian Diedrich;, “Deep Learning based Efficient Anomaly Detection for Securing Process Control Systems against Injection Attacks,” in *15th International Conference on Automation Science and Engineering (CASE)*, 2019.
- [114] K. Alrawashdeh and C. Purdy, “Toward an Online Anomaly Intrusion Detection System Based on Deep Learning,” in *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, pp. 195–200.
- [115] Jaspreet. Minhas. Harjinder Kaur, GUrpreet Singh, “A Review of Machine Learning based Anomaly Detection Techniques,” *Int. J. Comput. Appl. Technol. Res.*, vol. 2, no. 2, pp. 185–187, 2013.
- [116] Nguyen Thanh Van, Tran Ngoc Thinh, “An anomaly-based network intrusion detection system using Deep learning,” in *International Conference on System Science and Engineering (ICSSE)*, 2017.
- [117] Wei Yu, David Griffith, Linqiang Ge and and Nada. Golmie, “An integrated detection system against false data injection attacks in the Smart Grid,” *Secur. Commun. NETWORKS*, vol. 8, pp. 91–109, 2014.
- [118] S. Huang, C. Zhou, S. Yang, and Y. Qin, “Cyber-physical System Security for Networked,” vol. 12, no. December, pp. 567–578, 2015.
- [119] Zhonghua. Pang Fangyuan. Hou.; Yuguo. Zhou. Dehui. Sun. “Design of False Data Injection Attacks for Output Tracking Control of CARMA Systems,” in *International*

References

- Conference on Information and Automation*, 2015, no. August, pp. 1273–1277.
- [120] J. Rabatel, S. Bringay, and P. Poncelet, “Anomaly detection in monitoring sensor data for preventive maintenance,” *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7003–7015, 2011.
- [121] David. J. Hill, Barbara. S. Minsker, and Eyal. Amir, “Real-time Bayesian anomaly detection in streaming environmental data,” *Water Resour. Res.*, vol. 46, no. 4, pp. 1–16, 2010.
- [122] Sudhir. Kumar. Sahu. Manoranjan Pradhan, Sateesh Kumar Pradhan, “Anomaly Detection Using Artificial Neural Networks,” *Int. J. Eng. Sci. Emerg. Technol.*, vol. 2, no. 1, pp. 29–36, 2012.
- [123] S. Siripanadorn, “Anomaly detection using self-organizing map and wavelets in wireless sensor networks,” *Proc. 10th WSEAS ...*, pp. 291–297, 2010.
- [124] M. Computing, “A comprehensive survey of false data injection in smart grid Zhitao Guan *, Nan Sun , Yue Xu and,” vol. 8, no. 1, 2015.
- [125] D. Wang, X. Guan, T. Liu, Y. Gu, Y. Sun, and Y. Liu, “A Survey on Bad Data Injection Attack in Smart Grid.”
- [126] Zubair. A. Baig and Abdul-Raouf. Amoudi, “An Analysis of Smart Grid Attacks and Countermeasures,” vol. 8, no. 8, 2013.
- [127] A. Anwar, “Vulnerabilities of Smart Grid State Estimation against False Data Injection Attack Cyber Incidents in Different Sector,” in *Renewable Energy Integration, Green Energy and Technology*, 2014.
- [128] Mohammad. Esmalifalak, Lanchao. Liu, and Nam Nguyen and Rang Zheng, “Detecting Stealthy False Data Injection Using Machine Learning in Smart Grid,” pp. 1–9, 2014.
- [129] J. Hao, S. Member, R. J. Piechocki, and D. Kaleshi, “Sparse Malicious False Data Injection Attacks and Defense Mechanisms in Smart Grids,” vol. 3203, no. c, pp. 1–12, 2015.
- [130] Sasanka Potluri; Christian Diedrich; Girish Kumar Reddy Sangala, “Identifying false data injection attacks in industrial control systems using artificial neural networks,” in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8.
- [131] Shaunak Mishra, Yasser Shoukry, Nikhil Karamchandani, Suhas N Diggavi, and Paulo Tabuada, “Secure state estimation against sensor attacks in the presence of noise,” *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 1, pp. 49–59, 2016.
- [132] Yuan Chen, Soumya Kar, and Jose MF Moura, “Cyber physical attacks with control objectives and detection constraints,” in *55th Conference on Decision and Control*, 2016, pp. 1125–1130.
- [133] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo, “Attack detection and identification in cyber-physical systems,” *IEEE Trans. Automat. Contr.*, vol. 58, no. 11, pp. 2715–2729, 2013.
- [134] Fei Miao, Quanyan Zhu, Miroslav Pajic, and George J Pappas, “Coding schemes for

References

- securing cyber-physical systems against stealthy data injection attacks,” *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 1, pp. 106–117, 2016.
- [135] Miroslav Pajic, Insup Lee, and George J Pappas, “Attack-resilient state estimation for noisy dynamical systems,” *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 1, pp. 82–92, 2016.
- [136] Chensheng Liu, Jing Wu, Chengnian Long, and Yebin Wang, “Dynamic state recovery for cyber-physical systems under switching location attacks,” *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 1, pp. 14–22, 2016.
- [137] Kebina Manandhar, Xiaojun Cao, Fei Hu, and Yao Liu, “Detection of faults and attacks including false data injection attack in smart grid using Kalman filter,” *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 4, pp. 370–379, 2014.
- [138] Qie Hu, Dariush Fooladivanda, Young Hwan Chang, and Claire J Tomlin, “Secure state estimation and control for cyber security of the nonlinear power systems,” *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1310–1321, 2017.
- [139] Yilin Mo, Emanuele Garone, Alessandro Casavola, and Bruno Sinopoli, “False data injection attacks against state estimation in wireless sensor networks,” in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5967–5972.
- [140] Le Xie, Yilin Mo, and Bruno Sinopoli, “False data injection attacks in electricity markets,” in *First IEEE International Conference on Smart Grid Communications*, 2010, pp. 226–231.
- [141] Yilin Mo and Bruno Sinopoli, “Integrity attacks on cyber-physical systems,” in *1st international conference on High Confidence Networked Systems*, 2012, pp. 47–54.
- [142] Yilin Mo and Bruno Sinopoli, “False data injection attacks in control systems,” in *1st workshop on Secure Control Systems*, 2010, pp. 1–6.
- [143] S. Hettich, S. and Bay, “KDD Cup 1999 Data,” *The UCI KDD Archive*, 1999. [Online]. Available: <http://kdd.ics.uci.edu>.
- [144] Mahbod. Tavallaee, Ebrahim. Bagheri, Wei. Lu, and Ali. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” *IEEE Symp. Comput. Intell. Secur. Def. Appl. CISDA 2009*, no. Cisda, pp. 1–6, 2009.
- [145] L. Dhanabal and S. P. Shantharajah, “A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.
- [146] Canadian Institute for Cybersecurity, “NSL-KDD dataset,” *University of New Brunswick*. [Online]. Available: <https://www.unb.ca/cic/datasets/nsl.html>. [Accessed: 16-Jan-2021].
- [147] Nour Moustafa and Jill Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in *Military communications and information systems conference*, 2015, pp. 1–6.
- [148] Nour Moustafa and Jill Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” *Inf. Secur. J. A Glob. Perspect.*, vol. 25, no. 1–3, pp. 18–31, 2016.

References

- [149] Nour Moustafa and Jill Slay, “The UNSW-NB15 Dataset Description,” *UNSW Webpage*, 2018. [Online]. Available: <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>. [Accessed: 17-Sep-2019].
- [150] P. R. Lyman and C. Georgakis, “Plant-wide control of the Tennessee Eastman problem,” *Comput. Chem. Eng.*, vol. 19, no. 3, pp. 321–331, 1995.
- [151] J. J. Downs and E. F. Vogel, “A plant-wide industrial process control problem,” *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993.
- [152] Festo Didactic, “MPS® PA Compact-Workstation mit Füllstands-, Durchfluss-, Druck- und Temperaturregelstrecken.” [Online]. Available: <http://www.festo-didactic.com/de-de/lernsysteme/prozessautomation,regelungstechnik/compact-workstation/mps-pa-compact-workstation-mit-fuellstands-,durchfluss-,druck-und-temperaturregelstrecken.htm?fbid=ZGUuZGUuNTQ0LjEzLjE4Ljg4Mi40Mzc2>. [Accessed: 15-Sep-2017].
- [153] A. Famili, W. Shen, R. Weber, and E. Simoudis, “Data Preprocessing and Intelligent Data Analysis,” vol. 1, pp. 3–23, 1997.
- [154] Sasanka Potluri, Christian Diedrich “Deep Feature Extraction for Multi-class Intrusion Detection in Industrial Control Systems,” *Int. J. Comput. Theroy Eng.*, vol. 9, no. 5, pp. 374–379, 2017.
- [155] J. Siekmann and W. Wahlster, *Advanced Intelligent Computing Theories and Applications*. .
- [156] MaxStat, “Tools for Scientific data Analysis - Statistics.” .
- [157] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*. MIT Press, 2016.
- [158] David.E. Rumelhart, Geoffrey .E. Hinton, Ronald J Willioams “Learning Internal Representations by Error Propagation.pdf.” .
- [159] Geoffrey E. Hinton, “Deep belief networks,” *University of Toronto*, 2009. [Online]. Available: http://scholarpedia.org/article/Deep_belief_networks. [Accessed: 17-Sep-2019].
- [160] Jae. Won. Kim, “Classification with Deep Belief Networks.”, 2013
- [161] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar, “Applications of Convolutional Neural Networks,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 7, no. 5, pp. 2206–2215, 2016.
- [162] Yoshua Bengio, “Learning deep architectures for AI,” *Found. trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
- [163] Saurabh Yadav, “Weight Initialization Techniques in Neural Networks,” *Towards Data Science*, 2018. [Online]. Available: <https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78>. [Accessed: 17-Sep-2019].
- [164] Derrick. H. Nguyen and Bernard Widrow, “Neural networks for self-learning control systems,” *IEEE Control Syst. Mag.*, vol. 10, no. 3, pp. 18–23, 1990.
- [165] Xavier Glorot and Yoshua Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *13th International conference on artificial intelligence and*

-
- statistics*, 2010, pp. 249–256.
- [166] Chigozie Nwankpa, Winifred Ijomah, and Anthony Gachagan, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv Prepr. arXiv1811.03378*, 2018.
- [167] Frederic Bastien, Pascal Lamblin, and et. al., “Theano: new features and speed improvements,” *arXiv Prepr. arXiv1211.5590*, 2012.
- [168] Timothy Masters, *Practical neural network recipes in C++*. Morgan Kaufmann, 1993.
- [169] Andrew Y Ng, “Feature selection, L 1 vs. L 2 regularization, and rotational invariance,” in *21st International conference on Machine learning*, 2004, p. 78.
- [170] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, and et. al., “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [171] Sergey Ioffe and Christian Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv Prepr. arXiv1502.03167*, 2015.
- [172] Boris. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.
- [173] Nesterov Y, “A method of solving a convex programming problem with convergence rate $O(1/k^2)$,” *Sov. Math. Dokl*, vol. 27, 1983.
- [174] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, “On the importance of initialization and momentum in deep learning,” in *International conference on machine learning*, 2013, pp. 1139–1147.
- [175] Leon Bottou, “Stochastic gradient descent tricks,” in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 421–436.
- [176] Martin Riedmiller and Heinrich Braun, “RPROP-A fast adaptive learning algorithm,” in *Proc. of ISICIS VII*, *Universitat*, 1992.
- [177] John Duchi, Elad Hazan, and Yoram Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, 2011.
- [178] Matthew D Zeiler, “ADADELTA: an adaptive learning rate method,” *arXiv Prepr. arXiv1212.5701*, 2012.
- [179] Tijmen Tieleman and Geoffrey Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA Neural networks Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.
- [180] Diederik P Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv Prepr. arXiv1412.6980*, 2014.
- [181] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves, “Grid long short-term memory,” *arXiv Prepr. arXiv1507.01526*, 2015.

References

- [182] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Youshua Bengio, “Gated feedback recurrent neural networks,” in *International Conference on Machine Learning*, 2015, pp. 2067–2075.
- [183] Alex Graves, Greg Wayne, and Ivo Danihelka, “Neural turing machines,” *arXiv Prepr. arXiv1410.5401*, 2014.
- [184] Sasanka Potluri, “ICS-Injection_Attack_Toolbox,” *GitHub*, 2019. [Online]. Available: https://github.com/sasankapotluri/ICS-Injection_Attack_Toolbox. [Accessed: 05-Jun-2019].
- [185] Risidata, “CIA Trojan Causes Siberian Gas Pipeline Explosion.” [Online]. Available: <https://www.risidata.com/index.php?/Database/Detail/cia-trojan-causes-siberian-gas-pipeline-explosion>. [Accessed: 16-Sep-2019].
- [186] Alec Russell, “CIA plot led to huge blast in Siberian gas pipeline,” *The Telegraph*, 2004. [Online]. Available: <https://www.telegraph.co.uk/news/worldnews/northamerica/usa/1455559/CIA-plot-led-to-huge-blast-in-Siberian-gas-pipeline.html>. [Accessed: 16-Sep-2019].
- [187] Robert M. Lee, M. J. Assante, and T. Conway, “German steel mill cyber attack,” *Ind. Control Syst.*, vol. 30, p. 62, 2014.
- [188] Xiaobing He, “Threat Assessment for Multistage Cyber Attacks in Smart Grid Communication Networks,” University of Passau, 2017.
- [189] Oliver Schonschek and Peter Schmitz, “Cyber Kill Chain - Grundlagen, Anwendung und Entwicklung,” *Security Insider*, 2017. [Online]. Available: <https://www.security-insider.de/cyber-kill-chain-grundlagen-anwendung-und-entwicklung-a-608017/>. [Accessed: 16-Sep-2019].
- [190] BBC, “Ukraine power cut ‘was cyber-attack,’” *BBC Technology News*, 2017. [Online]. Available: <https://www.bbc.com/news/technology-38573074>. [Accessed: 16-Sep-2019].
- [191] Robert M. Lee, “Analysis of the Cyber attack on the Ukrainian Power Grid,” *Ind. Control Syst.*, p. 26, 2016.
- [192] Sasanka Potluri, “Automation Plant Use Case for Anomaly Detection.” [Online]. Available: <https://www.youtube.com/user/sasankapotluri>.
- [193] Coppeliarobotics, “Joint Types and Operations.” [Online]. Available: <http://www.coppeliarobotics.com/helpFiles/en/jointDescription.htm>. [Accessed: 02-Dec-2016].
- [194] Eurobots, “ABB IRB 2400 M94A.” [Online]. Available: http://www.eurobots.net/ABB_IRB-2400_M94A-en.html. [Accessed: 02-Dec-2016].
- [195] Kali Tools, “hping3 Package Description,” *Kali*. [Online]. Available: <https://tools.kali.org/information-gathering/hping3>. [Accessed: 17-Sep-2019].
- [196] NMAP.org, “Nmap Security Scanner,” *nmap*. [Online]. Available: <https://nmap.org/>. [Accessed: 17-Sep-2019].

References

- [197] angryip.org, “Angry IP Scanner.” [Online]. Available: <https://angryip.org/>. [Accessed: 17-Sep-2019].
- [198] Wireshark, “Wireshark.” [Online]. Available: <https://www.wireshark.org/>. [Accessed: 17-Sep-2019].
- [199] Arash Habibi Lashkari, Gerard Draper Gil, and Mohammad Saiful Islam Mamun, “ISCXFlowMeter,” *GitHub*. [Online]. Available: <https://github.com/ISCX/ISCXFlowMeter>. [Accessed: 17-Sep-2019].

Appendix A

Notable Cyber Attacks on Industrial Control Systems

Siberian Pipeline Explosion

A (Central Intelligence Agency) CIA to sabotage Soviet Industry by dumping Moscow into stealing booby-trapped software was spectacularly successfully when it triggered a huge explosion in a Siberian gas pipeline. The CIA inserted a hidden code into computer software illegally obtained by the Soviet Union. This software includes a Trojan Horse. The pipeline software that was to run the pumps, turbines, and valves was programmed to go haywire after a decent interval, to reset pump speeds and valve setting to produce pressures far beyond those acceptable to the pipeline joints and welds. The result was the most monumental non-nuclear explosion and fire ever seen from space in June 1982 [185] [186].

German Steel Mill Breach

"In December 2014, the German Government's Bundersamt für Sicherheit in der Informationstechnik (BSI) (translated as the Federal Office for Information Security) issued a report about a cyber attack on a steel mill that resulted in significant damage to the facility. The attack has received extensive publicity (from the BBC to YouTube) since then, while the technical details of the attack have been released by SANS [187] " [188].

"The BSI report stated that adversaries targeted industrial operators with spear phishing emails, which was observed in the HAVEX (targeting OPC communications) 1 and BlackEnergy Version 2 2 (targeting human-machine interface (HMI) products) malware threats. The attacker, described as an advanced persistent threat (APT) attacker, followed a pattern that is described as a "cyber kill chain" [189] to target the facility. At the first stage, the attacker sent out phishing emails to industrial operators and made use of social engineering techniques to gain access to the network. Those emails, which were attached with malicious documents (such as PDFs), once opened, executed malicious code that targeted an application vulnerability in the facility's corporate network. The attacker worked his/her way to the production network, i.e., industrial control system (ICS)). Owing to the connection between the corporate network and the production network, the exploitation of a vulnerability in the corporate network opened a remote connection point, allowing the attacker access to the production network. The second stage of the attack was the compromise of small sets of workstations. Once workstations were totally in his/her control, the attacker moved

into the plant network. Then, the attacker destroyed a blast furnace in the plant network by initiating its security settings in time, causing serious damage to the infrastructure. It took months to replace damaged equipment due to the need to remove and replace large pieces of machinery" [188].

Ukrainian Electric Disruption

"On 24, December 2015, TSN (a Ukrainian news outlet) released a report about power outages caused by a cyber attack 1. Numerous reporting agencies and independent bloggers, including the Washington Post, SANS Institute, New York Times, the BBC, CNN, Fox News, as well as the E-ISAC, had followed up on the initial TSN report and provided further details of that cyber attack, which was targeted at the Ukrainian electric system. The power outage caused by the cyber attack affected roughly 225,000 customers for over six hours during a spell of cold weather 2. Those outages were due to a combination of BlackEnergy Version 3, unreported backdoors, KillDisk, and malicious firmware uploads within the utility's systems. It was shown that the vulnerabilities in the utility (e.g., a lack of two-factor authentication, no resident capability to continually monitor the ICS network) had provided the adversary with the opportunity to persist within the environment for at least six months in order to conduct extensive reconnaissance and subsequently execute the attack 3. The attacks were conducted in three sophisticated, well-planned stages, as shown in Figure 1.2. During the cyber attack, spear phishing emails were sent out to gain access to the business networks of the three regional electric power distribution companies. The remote malicious opening of breakers in a number of substations was conducted by using either existing remote administration tools at the operating system level or remote ICS client software via virtual private network (VPN) connections. Modified KillDisk malware was used to erase selected files on targeted systems and corrupt the master boot record. The adversary also caused serial-to-Ethernet devices (located at substations) to malfunction at a firmware level. Moreover, the attacker also leveraged a remote telephonic denial of service on the energy company's call centre to ensure that the affected customers could not report outages and force the operator to move to a manual operation system in response to the attacks [190] [191]" [188].

The mentioned attacks are just a few but a lot of attacks still exist and every day a new attack is being identified. [9] gives an overview of the cyber-attack maps on ICS. These attacks with ability to compromise physical equipment are considered as the most trivial forms of attacks on ICS.

Appendix B

Complete Results

In this section, we can all obtained results in relation to the performance metrics of each and every individual algorithm in detail. A summary of these results is presented in Chapter 8 of this thesis.

B.1. NSL-KDD dataset

This sub-section provides the detailed overview about the performance of SAE and DBN on the NSL-KDD dataset. The summary of the attack classes and their attack types is mentioned in the Table 5-6. The results are self explanatory with the titles.

B.1.1 Stacked autoencoders

SAE – SMR (2-Classes)

Table B. 1: Accuracy for 2-class classification on NSL-KDD dataset with SAE+SMR

	Normal	Attack	Overall
Accuracy	89.97	92.70	91.24

Table B. 2: Performance metrics for 2-class classification on NSL-KDD dataset with SAE+SMR

Parameter	Normal	Attack
Precision	93.40	92.70
Recall	89.97	88.95
F-Measure	91.66	90.79

Appendix B

SAE – SMR (3-Classes)

Table B. 3: Accuracy for 3-class classification on NSL-KDD dataset with SAE+SMR

	Normal	DoS	Attack	Overall
Accuracy	88.95	96.18	76.19	90.30

Table B-4: Performance metrics for 3-class classification on NSL-KDD dataset with SAE+SMR

Parameter	Normal	Dos	Attack
Precision	93.04	93.33	82.45
Recall	92.92	96.13	74.08
F-Measure	92.98	94.71	78.04

SAE – SMR (4-Classes)

Table B-5: Accuracy for 4-class classification on NSL-KDD dataset with SAE+SMR

	Normal	DoS	Probe	Attack	Overall
Accuracy	85.90	97.46	78.65	91.97	89.49

Appendix B

Table B-6: Performance metrics for 4-class classification on NSL-KDD dataset with SAE+SMR

Parameter	Normal	Dos	Probe	Attack
Precision	95.39	94.84	74.87	16.35
Recall	85.90	97.46	78.68	91.97
F-Measure	90.40	96.13	79.73	27.77

SAE – SMR (5-Classes)

Table B-7: Accuracy for 5-class classification on NSL-KDD dataset with SAE+SMR

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	89.65	97.80	75.39	46.43	53.84	90.95

Table B-8: Performance metrics for 5-class classification on NSL-KDD dataset with SAE+SMR

Parameter	Normal	Dos	Probe	R2L	U2R
Precision	94.33	95.87	74.87	12.26	2.9
Recall	89.65	97.80	78.68	46.43	53.84
F-Measure	91.93	96.83	79.73	19.40	5.51

B.1.2 Deep belief networks

DBN – SMR (2-Classes)

Table B-9: Accuracy for 2-class classification on NSL-KDD dataset with DBN+SMR

	Normal	Attack	Overall
Accuracy	88.34	94.20	91.07

Table B-10: Performance metrics for 2-class classification on NSL-KDD dataset with DBN+SMR

Parameter	Normal	Attack
Precision	94.59	94.19
Recall	88.34	87.55
F-Measure	91.36	90.75

DBN – SMR (3-Classes)

Table B-11: Accuracy for 3-class classification on NSL-KDD dataset with DBN+SMR

	Normal	DoS	Attack	Overall
Accuracy	90.64	95.50	76.23	91.09

Appendix B

Table B-12: Performance metrics for 3-class classification on NSL-KDD dataset with DBN+SMR

Parameter	Normal	Dos	Attack
Precision	92.99	97.40	63.30
Recall	90.64	95.50	76.23
F-Measure	91.80	96.44	69.16

DBN – SMR (4-Classes)

Table B-13: Accuracy for 4-class classification on NSL-KDD dataset with DBN+SMR

	Normal	DoS	Probe	Attack	Overall
Accuracy	88.30	95.83	83.97	86.53	90.63

Table B-14: Performance metrics for 4-class classification on NSL-KDD dataset with DBN+SMR

Parameter	Normal	Dos	Probe	Attack
Precision	95.33	95.83	76.67	17.84
Recall	88.30	96.19	83.97	86.53
F-Measure	91.68	96.01	80.16	29.59

DBN – SMR (5-Classes)

Table B-15: Accuracy for 5-class classification on NSL-KDD dataset with DBN+SMR

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	89.18	96.06	83.81	83.91	15.38	91.14

Table B-16: Performance metrics for 5-class classification on NSL-KDD dataset with DBN+SMR

Parameter	Normal	Dos	Probe	R2L	U2R
Precision	95.04	96.41	76.43	23.58	1.15
Recall	89.18	96.06	83.81	83.91	15.38
F-Measure	92.02	96.24	79.95	36.82	2.15

B.2. UNSW-NB-15 dataset

This sub-section provides the performance results of SAE and DBN on the UNSW-NB-15 dataset.

B.2.1 Stacked autoencoders

SAE – SMR (2-Classes)

Table B-17: Accuracy for 2-class classification on UNS-NB-15 dataset with SAE+SMR

	Normal	Attack	Overall
Accuracy	94.62	90.70	92.54

Appendix B

Table B-18: Performance metrics for 2-class classification on UNS-NB-15 dataset with SAE+SMR

Parameter	Normal	Attack
Precision	95.40	90.91
Recall	94.84	89.93
F-Measure	93.62	91.24

SAE – SMR (3-Classes)

Table B-19: Accuracy for 3-class classification on UNS-NB-15 dataset with SAE+SMR

	Normal	Generic	Attack	Overall
Accuracy	93.86	96.68	91.46	94.68

Table B-20: Performance metrics for 3-class classification on UNS-NB-15 dataset with SAE+SMR

Parameter	Normal	Generic	Attack
Precision	95.47	97.26	70.24
Recall	93.73	96.57	90.89
F-Measure	94.64	96.94	80.43

SAE – SMR (4-Classes)

Table B-21: Accuracy for 4-class classification on UNS-NB-15 dataset with SAE+SMR

	Normal	Generic	Exploit	Attack	Overall
Accuracy	94.93	97.30	69.10	72.80	94.86

Table B-22: Performance metrics for 4-class classification on UNS-NB-15 dataset with SAE+SMR

Parameter	Normal	Generic	Exploit	Attack
Precision	93.85	98.62	81.48	43.26
Recall	94.97	96.72	67.81	72.46
F-Measure	93.68	97.92	69.42	56.53

SAE – SMR (5-Classes)

Table B-23: Accuracy for 5-class classification on UNS-NB-15 dataset with SAE+SMR

	Normal	Generic	Exploits	Fuzzers	Attack	Overall
Accuracy	95.70	97.34	70.46	68.43	69.52	94.96

Appendix B

Table B-24: Performance metrics for 5-class classification on UNS-NB-15 dataset with SAE+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	Attack
Precision	96.48	96.56	69.82	65.28	51.22
Recall	94.53	98.27	71.62	70.32	63.54
F-Measure	97.82	97.43	70.98	69.72	48.35

SDA – SMR (6-Classes)

Table B-25: Accuracy for 6-class classification on UNS-NB-15 dataset with SAE+SMR

	Normal	Generic	Exploits	Fuzzers	DoS	Attack	Overall
Accuracy	95.65	97.28	78.20	70.43	64.67	58.48	93.62

Table B-26: Performance metrics for 6-class classification on UNS-NB-15 dataset with SAE+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	DoS	Attack
Precision	96.79	96.46	72.53	54.42	51.28	49.43
Recall	95.45	97.38	77.32	68.35	63.94	59.25
F-Measure	97.23	98.24	79.74	69.86	65.29	57.65

Appendix B

SDA – SMR (7-Classes)

Table B-27: Accuracy for 7-class classification on UNS-NB-15 dataset with SAE+SMR

	Normal	Generic	Exploits	Fuzzers	DoS	Reconnaissance	Attack	Overall
Accuracy	95.62	97.19	77.93	69.85	58.64	48.28	22.14	93.20

Table B-28: Performance metrics for 7-class classification on UNS-NB-15 dataset with SAE+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	DoS	Reconnaissance	Attack
Precision	96.43	96.38	71.85	55.39	47.26	35.68	11.44
Recall	95.29	97.42	77.24	69.46	56.49	47.16	20.62
F-Measure	96.94	98.14	78.92	68.73	45.34	37.21	17.37

Table B-29: Accuracy for 8-class classification on UNS-NB-15 dataset with SAE+SMR

	Accuracy
Normal	95.68
Generic	97.24
Exploits	77.25
Fuzzers	69.38
DoS	56.16
Reconnaissance	43.82
Analysis	10.47
Attack	18.58
Overall	93.41

Appendix B

Table B-30: Performance metrics for 8-class classification on UNS-NB-15 dataset with SAE+SMR

	Precision	Recall	F-Measure
Normal	92.32	95.03	93.16
Generic	93.27	96.87	96.56
Exploits	70.83	76.69	72.49
Fuzzers	55.08	67.81	68.28
DoS	35.13	54.07	54.22
Reconnaissance	31.43	42.91	44.35
Analysis	20.50	10.89	09.14
Attack	10.47	17.60	16.04

Table B-31: Accuracy for 9-class classification on UNS-NB-15 dataset with SAE+SMR

	Accuracy
Normal	95.62
Generic	97.21
Exploits	77.19
Fuzzers	69.29
DoS	54.23
Reconnaissance	41.64
Analysis	10.27
Backdoor	4.71
Attack	2.14
Overall	93.28

Appendix B

Table B-32: Performance metrics for 9-class classification on UNS-NB-15 dataset with SAE+SMR

	Precision	Recall	F-Measure
Normal	92.26	95.24	93.43
Generic	93.34	96.92	95.94
Exploits	70.68	76.62	75.61
Fuzzers	55.54	66.95	67.08
DoS	33.81	53.72	52.44
Reconnaissance	28.93	40.89	42.11
Analysis	18.06	10.48	10.69
Backdoor	8.63	3.86	4.33
Attack	5.47	2.89	2.04

Appendix B

SDA – SMR (10-Classes)

Table B-33: Accuracy for 10-class classification on UNS-NB-15 dataset with SAE+SMR

	Accuracy
Normal	95.60
Generic	97.19
Exploits	77.14
Fuzzers	69.21
DoS	54.18
Reconnaissance	41.24
Analysis	10.25
Backdoor	4.40
Shellcode	1.12
Worms	0
Overall	93.17

Appendix B

Table B-34: Performance metrics for 10-class classification on UNS-NB-15 dataset with SAE+SMR

	Precision	Recall	F-Measure
Normal	91.87	95.43	94.16
Generic	92.91	96.74	96.21
Exploits	70.27	75.97	76.24
Fuzzers	53.34	65.63	68.13
DoS	31.52	53.60	51.09
Reconnaissance	26.82	38.73	40.28
Analysis	16.64	10.33	10.49
Backdoor	6.71	3.94	4.22
Shellcode	4.03	1.28	1.08
Worms	0	0	0

B.2.2 Deep belief networks

DBN – SMR (2-Classes)

Table B-35: Accuracy for 2-class classification on UNS-NB-15 dataset with DBN+SMR

	Normal	Attack	Overall
Accuracy	91.14	90.82	91.15

Table B-36: Performance metrics for 2-class classification on UNS-NB-15 dataset with DBN+SMR

Parameter	Normal	Attack
Precision	94.82	93.69
Recall	90.78	89.38
F-Measure	93.42	90.92

DBN – SMR (3-Classes)

Table B-37: Accuracy for 3-class classification on UNS-NB-15 dataset with DBN+SMR

	Normal	Generic	Attack	Overall
Accuracy	92.48	95.57	90.82	93.63

Appendix B

Table B-38: Performance metrics for 3-class classification on UNS-NB-15 dataset with DBN+SMR

Parameter	Normal	Generic	Attack
Precision	96.24	94.75	71.32
Recall	91.86	95.22	91.37
F-Measure	92.51	94.83	82.64

DBN – SMR (4-Classes)

Table B-39: Accuracy for 4-class classification on UNS-NB-15 dataset with DBN+SMR

	Normal	Generic	Exploit	Attack	Overall
Accuracy	93.21	95.40	64.42	68.76	94.18

Table B-40: Performance metrics for 4-class classification on UNS-NB-15 dataset with DBN+SMR

Parameter	Normal	Generic	Exploit	Attack
Precision	94.63	95.28	85.51	54.27
Recall	93.18	94.62	67.94	67.30
F-Measure	94.36	93.79	68.29	68.15

Appendix B

DBN – SMR (5-Classes)

Table B-41: Accuracy for 5-class classification on UNS-NB-15 dataset with DBN+SMR

	Normal	Generic	Exploits	Fuzzers	Attack	Overall
Accuracy	92.82	95.63	66.41	60.54	58.29	92.93

Table B-42: Performance metrics for 5-class classification on UNS-NB-15 dataset with DBN+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	Attack
Precision	94.81	95.64	58.56	64.48	54.54
Recall	93.26	96.23	59.88	59.71	56.93
F-Measure	94.48	95.48	60.27	60.55	57.58

DBN – SMR (6-Classes)

Table B-43: Accuracy for 6-class classification on UNS-NB-15 dataset with DBN+SMR

	Normal	Generic	Exploits	Fuzzers	DoS	Attack	Overall
Accuracy	91.87	95.28	64.74	60.38	58.44	48.62	91.43

Appendix B

Table B-44: Performance metrics for 6-class classification on UNS-NB-15 dataset with DBN+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	DoS	Attack
Precision	95.48	94.26	77.59	59.82	52.29	50.41
Recall	91.62	96.12	63.87	59.66	60.08	47.90
F-Measure	90.84	95.68	64.64	60.75	59.54	49.22

DBN – SMR (7-Classes)

Table B-45: Accuracy for 7-class classification on UNS-NB-15 dataset with DBN+SMR

	Normal	Generic	Exploits	Fuzzers	DoS	Reconnaissance	Attack	Overall
Accuracy	90.52	94.41	61.85	59.67	57.63	44.93	16.11	90.87

Table B-46: Performance metrics for 7-class classification on UNS-NB-15 dataset with DBN+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	DoS	Reconnaissance	Attack
Precision	94.22	94.21	74.43	60.34	52.84	38.72	18.62
Recall	90.71	93.84	62.54	58.76	58.07	43.46	15.67
F-Measure	90.69	94.68	61.34	59.19	57.80	44.87	17.71

Appendix B

DBN – SMR (8-Classes)

Table B-47: Accuracy for 8-class classification on UNS-NB-15 dataset with DBN+SMR

	Accuracy
Normal	90.46
Generic	94.19
Exploits	61.79
Fuzzers	58.23
DoS	56.47
Reconnaissance	43.76
Analysis	0
Attack	0
Overall	90.14

Appendix B

Table B-48: Performance metrics for 8-class classification on UNS-NB-15 dataset with DBN+SMR

	Precision	Recall	F-Measure
Normal	94.12	91.18	90.82
Generic	94.17	93.75	94.51
Exploits	72.28	61.43	60.94
Fuzzers	60.28	59.56	58.86
DoS	51.64	56.27	55.19
Reconnaissance	35.61	41.93	42.81
Analysis	0	0	0
Attack	0	0	0

Appendix B

DBN – SMR (9-Classes)

Table B-49: Accuracy for 9-class classification on UNS-NB-15 dataset with DBN+SMR

	Accuracy
Normal	90.38
Generic	94.08
Exploits	61.64
Fuzzers	57.14
DoS	55.86
Reconnaissance	41.97
Analysis	0
Backdoor	0
Attack	0
Overall	90.02

Appendix B

Table B-50: Performance metrics for 9-class classification on UNS-NB-15 dataset with DBN+SMR

	Precision	Recall	F-Measure
Normal	94.09	91.12	90.79
Generic	94.08	93.94	94.46
Exploits	72.87	61.57	61.22
Fuzzers	59.42	58.06	57.51
DoS	51.26	56.17	55.13
Reconnaissance	34.49	41.73	41.97
Analysis	0	0	0
Backdoor	0	0	0
Attack	0	0	0

Appendix B

DBN – SMR (10-Classes)

Table B-51: Accuracy for 10-class classification on UNS-NB-15 dataset with DBN+SMR

	Accuracy
Normal	90.32
Generic	94.01
Exploits	61.48
Fuzzers	57.71
DoS	56.21
Reconnaissance	42.15
Analysis	0
Backdoor	0
Shellcode	0
Worms	0
Overall	89.97

Appendix B

Table B-52: Performance metrics for 10-class classification on UNS-NB-15 dataset with DBN+SMR

	Precision	Recall	F-Measure
Normal	94.06	91.08	90.67
Generic	93.74	94.12	94.38
Exploits	71.52	61.47	61.36
Fuzzers	58.24	58.45	57.62
DoS	50.83	56.27	55.09
Reconnaissance	32.75	41.63	41.81
Analysis	0	0	0
Backdoor	0	0	0
Shellcode	0	0	0
Worms	0	0	0

B.3. Injection attack dataset

This sub-section provides the performance results of SAE and DBN on the injection attacks dataset.

B.3.1 Stacked autoencoders

Level sensor data – Measurement injection

Table B-53: Accuracy for measurement injection attacks with SAE+SMR

	Normal	Attack	Overall
Accuracy	97.44	96.69	97.18

Table B-54: Performance metrics for measurement injection attacks with SAE+SMR

Parameter	Normal	Attack
Precision	98.26	95.14
Recall	97.44	96.69
F-Measure	97.85	95.91

Pump data – Command injection

Table B-55: Accuracy for command injection attacks with SAE+SMR

	Normal	Attack	Overall
Accuracy	97.07	94.11	96.09

Table B-56: Performance metrics for command injection attacks with SAE+SMR

Parameter	Normal	Attack
Precision	97.10	94.05
Recall	97.07	94.11
F-Measure	97.08	94.08

B.3.2 Deep belief networks

Level sensor data – Measurement injection

Table B-57: Accuracy for measurement injection attacks with DBN+SMR

	Normal	Attack	Overall
Accuracy	86.39	88.51	87.11

Table B-58: Performance metrics for measurement injection attacks with DBN+SMR

Parameter	Normal	Attack
Precision	93.54	77.14
Recall	86.39	88.51
F-Measure	89.82	82.44

Pump data – Command injection

Table B-59: Accuracy for command injection attacks with DBN+SMR

	Normal	Attack	Overall
Accuracy	97.45	57.05	84.13

Table B-60: Performance metrics for command injection attacks with DBN+SMR

Parameter	Normal	Attack
Precision	82.18	91.68
Recall	97.45	57.05
F-Measure	89.17	70.33

B.4. Convolutional neural networks

This sub-section provides the performance results of CNN on NSL-KDD, UNSW-NB-15 and injection attacks dataset.

B.4.1 NSL-KDD dataset

CNN – SMR (2-Classes)

Table B-61: Accuracy for 2-class classification on NSL-KDD dataset with CNN+SMR

	Normal	Attack	Overall
Accuracy	89.97	92.70	91.24

Table B-62: Performance metrics for 2-class classification on NSL-KDD dataset with CNN+SMR

Parameter	Normal	Attack
Precision	93.40	92.70
Recall	89.97	88.95
F-Measure	91.66	90.79

CNN – SMR (3-Classes)

Table B-63: Accuracy for 3-class classification on NSL-KDD dataset with CNN+SMR

	Normal	DoS	Attack	Overall
Accuracy	88.95	96.18	76.19	90.30

Appendix B

Table B-64: Performance metrics for 3-class classification on NSL-KDD dataset with CNN+SMR

Parameter	Normal	Dos	Attack
Precision	93.04	93.33	82.45
Recall	92.92	96.13	74.08
F-Measure	92.98	94.71	78.04

CNN – SMR (4-Classes)

Table B-65: Accuracy for 4-class classification on NSL-KDD dataset with CNN+SMR

	Normal	DoS	Probe	Attack	Overall
Accuracy	85.90	97.46	78.65	91.97	89.49

Table B-66: Performance metrics for 4-class classification on NSL-KDD dataset with CNN+SMR

Parameter	Normal	Dos	Probe	Attack
Precision	95.39	94.84	74.87	16.35
Recall	85.90	97.46	78.68	91.97
F-Measure	90.40	96.13	79.73	27.77

Appendix B

CNN – SMR (5-Classes)

Table B-67: Accuracy for 5-class classification on NSL-KDD dataset with CNN+SMR

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	89.65	97.80	75.39	46.43	53.84	90.95

Table B-68: Performance metrics for 5-class classification on NSL-KDD dataset with CNN+SMR

Parameter	Normal	Dos	Probe	R2L	U2R
Precision	94.33	95.87	74.87	12.26	2.9
Recall	89.65	97.80	78.68	46.43	53.84
F-Measure	91.93	96.83	79.73	19.40	5.51

B.4.2 UNSW-NB-15 dataset

CNN – SMR (2-Classes)

Table B-69: Accuracy for 2-class classification on UNS-NB-15 dataset with CNN+SMR

	Normal	Attack	Overall
Accuracy	99.10	93.70	98.00

Appendix B

Table B-70: Performance metrics for 2-class classification on UNS-NB-15 dataset with CNN+SMR

Parameter	Normal	Attack
Precision	98.41	96.46
Recall	99.13	93.66
F-Measure	98.77	95.04

CNN – SMR (3-Classes)

Table B-71: Accuracy for 3-class classification on UNS-NB-15 dataset with CNN+SMR

	Normal	Generic	Attack	Overall
Accuracy	97.80	97.70	97.90	97.80

Table B-72: Performance metrics for 3-class classification on UNS-NB-15 dataset with CNN+SMR

Parameter	Normal	Generic	Attack
Precision	99.93	99.45	74.14
Recall	97.77	97.69	97.92
F-Measure	98.84	98.56	84.39

Appendix B

CNN – SMR (4-Classes)

Table B-73: Accuracy for 4-class classification on UNS-NB-15 dataset with CNN+SMR

	Normal	Generic	Exploit	Attack	Overall
Accuracy	99.20	97.70	39.00	64.30	96.20

Table B-74: Performance metrics for 4-class classification on UNS-NB-15 dataset with CNN+SMR

Parameter	Normal	Generic	Exploit	Attack
Precision	98.32	99.46	68.76	54.10
Recall	99.19	97.69	39.01	64.35
F-Measure	98.75	98.57	49.78	58.78

CNN – SMR (5-Classes)

Table B-75: Accuracy for 5-class classification on UNS-NB-15 dataset with CNN+SMR

	Normal	Generic	Exploits	Fuzzers	Attack	Overall
Accuracy	99.70	97.70	56.30	0	12.30	95.00

Appendix B

Table B-76: Performance metrics for 5-class classification on UNS-NB-15 dataset with CNN+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	Attack
Precision	95.86	99.46	55.77	0	49.19
Recall	99.68	97.69	56.31	0	12.33
F-Measure	97.73	98.57	56.04	0	19.72

CNN – SMR (6-Classes)

Table B-77: Accuracy for 6-class classification on UNS-NB-15 dataset with CNN+SMR

	Normal	Generic	Exploits	Fuzzers	DoS	Attack	Overall
Accuracy	99.70	97.70	62.20	0.60	0	0	94.90

Table B-78: Performance metrics for 6-class classification on UNS-NB-15 dataset with CNN+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	DoS	Attack
Precision	95.79	99.46	53.73	5.13	0	0
Recall	99.68	97.69	62.22	0.56	0	0
F-Measure	97.70	98.57	57.66	1.01	0	0

Appendix B

CNN – SMR (7-Classes)

Table B-79: Accuracy for 7-class classification on UNS-NB-15 dataset with CNN+SMR

	Normal	Generic	Exploits	Fuzzers	DoS	Reconnaissance	Attack	Overall
Accuracy	99.70	97.70	62.10	0.40	0	0	0	94.90

Table B-80: Performance metrics for 7-class classification on UNS-NB-15 dataset with CNN+SMR

Parameter	Normal	Generic	Exploits	Fuzzers	DoS	Reconnaissance	Attack
Precision	95.72	99.46	54.05	4.66	0	0	0
Recall	99.68	97.69	62.14	0.37	0	0	0
F-Measure	97.66	98.57	57.81	0.69	0	0	0

CNN – SMR (8-Classes)

Table B-81: Accuracy for 8-class classification on UNS-NB-15 dataset with CNN+SMR

	Accuracy
Normal	99.70
Generic	97.70
Exploits	61.40
Fuzzers	0
DoS	0
Reconnaissance	0
Analysis	0
Attack	0
Overall	94.90

Appendix B

Table B-82: Performance metrics for 8-class classification on UNS-NB-15 dataset with CNN+SMR

	Precision	Recall	F-Measure
Normal	95.56	99.69	97.58
Generic	99.42	97.69	98.55
Exploits	54.09	61.42	57.52
Fuzzers	0	0	0
DoS	0	0	0
Reconnaissance	0	0	0
Analysis	0	0	0
Attack	0	0	0

CNN – SMR (9-Classes)

Table B-83: Accuracy for 9-class classification on UNS-NB-15 dataset with CNN+SMR

	Accuracy
Normal	99.70
Generic	97.70
Exploits	61.60
Fuzzers	0
DoS	0
Reconnaissance	0
Analysis	0
Backdoor	0
Attack	0
Overall	94.90

Appendix B

Table B-84: Performance metrics for 9-class classification on UNS-NB-15 dataset with CNN+SMR

	Precision	Recall	F-Measure
Normal	95.59	99.69	97.60
Generic	99.42	97.69	98.55
Exploits	54.06	61.59	57.58
Fuzzers	0	0	0
DoS	0	0	0
Reconnaissance	0	0	0
Analysis	0	0	0
Backdoor	0	0	0
Attack	0	0	0

Appendix B

CNN – SMR (10-Classes)

Table B-85: Accuracy for 10-class classification on UNS-NB-15 dataset with CNN+SMR

	Accuracy
Normal	99.70
Generic	97.70
Exploits	61.80
Fuzzers	0
DoS	0
Reconnaissance	0
Analysis	0
Backdoor	0
Shellcode	0
Worms	0
Overall	94.90

Appendix B

Table B-86: Performance metrics for 10-class classification on UNS-NB-15 dataset with CNN+SMR

	Precision	Recall	F-Measure
Normal	95.61	99.69	97.61
Generic	99.42	97.69	98.55
Exploits	53.97	61.85	57.64
Fuzzers	0	0	0
DoS	0	0	0
Reconnaissance	0	0	0
Analysis	0	0	0
Backdoor	0	0	0
Shellcode	0	0	0
Worms	0	0	0

B.4.3. Injection attack dataset

Level sensor data – Measurement injection

Table B-87: Accuracy for measurement injection attacks with CNN+SMR

	Normal	Attack	Overall
Accuracy	97.50	94.90	96.70

Table B-88: Performance metrics for measurement injection attacks with CNN+SMR

Parameter	Normal	Attack
Precision	97.74	94.37
Recall	97.52	94.86
F-Measure	97.63	94.61

Pump data – Command injection

Table B-89: Accuracy for command injection attacks with CNN+SMR

	Normal	Attack	Overall
Accuracy	95.90	92.70	94.90

Table B-90: Performance metrics for command injection attacks with CNN+SMR

Parameter	Normal	Attack
Precision	96.90	90.40
Recall	95.87	92.69
F-Measure	96.38	91.53

B.5. Hybrid deep learning

The implementation of hybrid deep learning algorithms was implemented using only NSL-KDD dataset with SAE and DBN algorithms. This sub-section provides the results of the provided hybrid deep learning algorithms mentioned in Section. 7.5.

SAE – SVM (2-Classes)

TableB-91: Accuracy for 2-class classification on NSL-KDD dataset with SAE+SVM

	Normal	Attack	Overall
Accuracy	90.07	92.70	91.63

Appendix B

Table B-92: Performance metrics for 2-class classification on NSL-KDD dataset with SAE+SVM

Parameter	Normal	Attack
Precision	94.02	93.43
Recall	90.07	89.12
F-Measure	92.00	91.22

SAE – SVM (3-Classes)

Table B-93: Accuracy for 3-class classification on NSL-KDD dataset with SAE+SVM

	Normal	DoS	Attack	Overall
Accuracy	92.04	95.50	80.24	92.11

Table B-94: Performance metrics for 3-class classification on NSL-KDD dataset with SAE+SVM

Parameter	Normal	Dos	Attack
Precision	92.37	98.21	67.09
Recall	92.89	95.48	71.80
F-Measure	92.63	96.82	69.39

SAE – SVM (4-Classes)

Table B-95: Accuracy for 4-class classification on NSL-KDD dataset with SAE+SVM

	Normal	DoS	Probe	Attack	Overall
Accuracy	91.26	95.24	72.41	86.15	90.93

Table B-96: Performance metrics for 4-class classification on NSL-KDD dataset with SAE+SVM

Parameter	Normal	Dos	Probe	Attack
Precision	93.04	95.24	80.39	25.84
Recall	91.26	95.24	72.41	86.15
F-Measure	92.14	95.24	76.19	39.76

Appendix B

SAE – SVM (5-Classes)

Table B-97: Accuracy for 5-class classification on NSL-KDD dataset with SAE+SVM

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	91.47	96.05	78.60	65.72	19.23	91.71

TableB-98: Performance metrics for 5-class classification on NSL-KDD dataset with SAE+SVM

Parameter	Normal	Dos	Probe	R2L	U2R
Precision	93.44	95.06	87.16	21.38	6.66
Recall	91.47	96.05	78.60	65.72	9.61
F-Measure	92.44	95.55	82.66	32.27	7.87

DBN – SVM (2-Classes)

Table B-99: Accuracy for 2-class classification on NSL-KDD dataset with DBN+SVM

	Normal	Attack	Overall
Accuracy	90.04	92.76	91.31

Appendix B

Table B-100: Performance metrics for 2-class classification on NSL-KDD dataset with DBN+SVM

Parameter	Normal	Attack
Precision	93.46	92.76
Recall	90.04	89.03
F-Measure	91.72	90.86

DBN – SVM (3-Classes)

Table B-101: Accuracy for 3-class classification on NSL-KDD dataset with DBN+SVM

	Normal	DoS	Attack	Overall
Accuracy	91.38	95.60	81.60	91.93

Table B-102: Performance metrics for 3-class classification on NSL-KDD dataset with DBN+SVM

Parameter	Normal	Dos	Attack
Precision	93.40	98.06	66.08
Recall	91.38	95.60	81.60
F-Measure	92.64	96.81	73.02

DBN – SVM (4-Classes)

Table B-103: Accuracy for 4-class classification on NSL-KDD dataset with DBN+SVM

	Normal	DoS	Probe	Attack	Overall
Accuracy	89.93	96.54	74.16	78.70	90.73

TableB-104: Performance metrics for 4-class classification on NSL-KDD dataset with DBN+SVM

Parameter	Normal	Dos	Probe	Attack
Precision	94.53	96.54	75.57	20.96
Recall	89.83	95.13	74.16	78.70
F-Measure	92.12	95.83	74.86	33.10

DBN – SVM (5-Classes)

Table B-105: Accuracy for 5-class classification on NSL-KDD dataset with DBN+SVM

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	92.06	95.33	75.19	35.57	46.15	91.23

Table B-106: Performance metrics for 5-class classification on NSL-KDD dataset with DBN+SVM

Parameter	Normal	Dos	Probe	R2L	U2R
Precision	92.75	96.55	78.35	14.08	29.26
Recall	92.06	95.33	75.19	35.57	46.15
F-Measure	92.40	95.94	76.74	20.17	35.82

After evaluating the above-mentioned hybrid deep learning algorithms, other combinations such as SAE+DBN+SMR and SAE+DBN+SVM didn't provide significant improvements in detection accuracies. Due to this reason, further evaluation of performance metrics such as precision, recall and F-measure weren't performed. The detection accuracies are provided below for different classes as a reference.

Appendix B

SAE+DBN – SMR (2-Classes)

Table B-107: Accuracy for 2-class classification on NSL-KDD dataset with SAE+DBN+SMR

	Normal	Attack	Overall
Accuracy	91.33	91.81	91.55

SAE+DBN – SMR (3-Classes)

Table B-108: Accuracy for 3-class classification on NSL-KDD dataset with SAE+DBN+SMR

	Normal	DoS	Attack	Overall
Accuracy	91.78	95.73	68.37	90.86

SAE+DBN – SMR (4-Classes)

Table B-109: Accuracy for 4-class classification on NSL-KDD dataset with SAE+DBN+SMR

	Normal	DoS	Probe	Attack	Overall
Accuracy	89.56	95.64	59.42	41.30	88.59

Appendix B

SAE+DBN – SMR (5-Classes)

Table B-110: Accuracy for 5-class classification on NSL-KDD dataset with SAE+DBN+SMR

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	90.56	95.79	78.21	49.74	21.11	90.92

SAE+DBN – SVM (2-Classes)

Table B-111: Accuracy for 2-class classification on NSL-KDD dataset with SAE+DBN+ SVM

	Normal	Attack	Overall
Accuracy	91.46	92.23	91.82

SAE+DBN – SVM (3-Classes)

Table B-112: Accuracy for 3-class classification on NSL-KDD dataset with SAE+DBN+SVM

	Normal	DoS	Attack	Overall
Accuracy	92.55	95.02	77.82	91.97

Appendix B

SAE+DBN – SVM (4-Classes)

Table B-113: Accuracy for 4-class classification on NSL-KDD dataset with SAE+DBN+ SVM

	Normal	DoS	Probe	Attack	Overall
Accuracy	92.34	95.57	73.58	82,21	91.69

SAE+DBN – SVM (5-Classes)

Table B-114: Accuracy for 5-class classification on NSL-KDD dataset with SAE+DBN+ SVM

	Normal	DoS	Probe	R2L	U2R	Overall
Accuracy	89.90	95.36	81.03	86.43	44.23	91.02

Appendix C

Shallow Neural Networks for Injection Attacks in automation applications

"A screenshot of the developed use case is shown in the Fig. 72. In this use case, two robotic arms are simulated. One, ABB IRB 140 for pick and place functions and second, KUKA LBR 4+ for pushing functions. We have two conveyor belts to move different sized packets in the environment. An Omni-directional platform is used for the purpose of transporting the finished product to a different place in the plant. Along with robots there exist different sensors (proximity sensors, visual sensors) which provide the information about the status of the plant. ABB robot has to pick a small box from one conveyor belt and place it into a big box on another conveyor belt. After placing certain number of small boxes into the big box the KUKA robot has to push the big box so that the filled box passes over a conveyor belt and reaches the Omni-directional platform and it transports the collected boxes to the required location in the plant" [130].

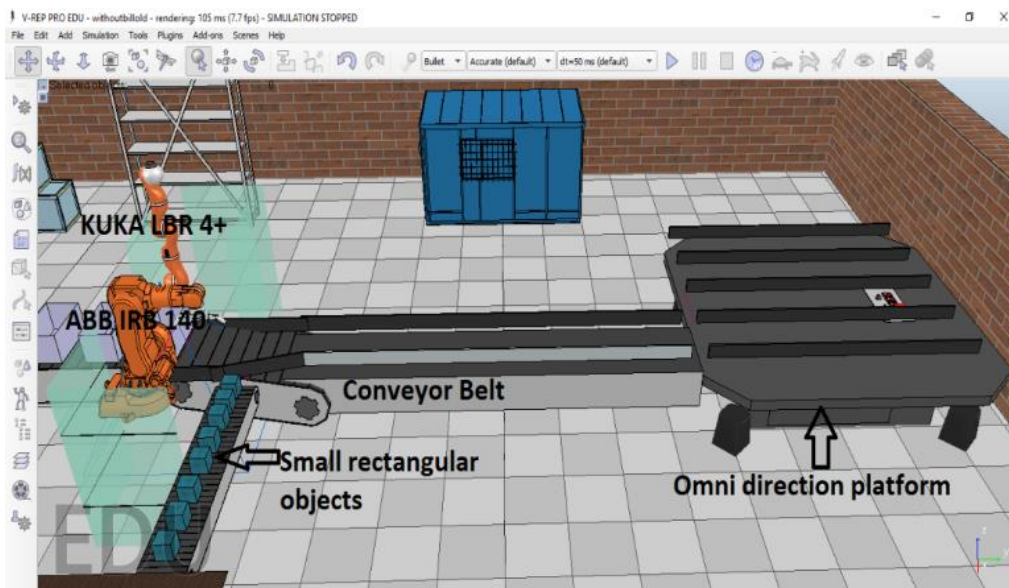


Figure 72: Manufacturing plant use-case in V-REP

"All the above-mentioned actions need to happen with high accuracy in a periodic manner to fulfil the purpose of the plant. A small disturbance in the plant either on the robot's side or on the process side will lead to huge damage or dangerous situation" [130].

"From the simulation, we continuously acquired the position information and the joints information of the robots and different sensors data was used to report the status of the plant. The streamed data

from the plant is connected to MATLAB/Simulink via remote API interface. The remote API interface translates the V-REP data so that MATLAB can interpret it. The obtained information is used to give the control commands back to V-REP. A control algorithm is developed in Simulink to make the plant run based on the desired functionality. The choice of Simulink based control algorithm eases the further integration and testing of the trained neural network for attack detection into the plant control loop. The simulated V-REP videos can be seen in [192]" [130].

False Data Injection Tool

"False data injection tool plays a key role in both training as well as testing phase. For this paper, we created two varieties of injection tools. One is a MATLAB function which takes the data from workspace where the data from the plant is stored and the other is a Graphical User Interface (GUI) based injection tool using MATLAB GUIDE (Graphical User Interface Design Environment)" [130].

"During training phase, the MATLAB function injects different types of attacks into the data set. The attacks injected into the data set will disturb the position of the robots, the packets information from visual sensor and the control commands given to the robots. The attacks on the position of the robots causes a misinterpretation in the arms position which make the Simulink to choose the false actuation command and will disturb the entire functionality of the plant. The visual sensor provides the information about the number of packets placed in the big box. Attacks on this information initiates the wrong actuation of the KUKA robot which pushes the boxes before getting filled. In forward kinematic robotics control, the movement of individual joint in a specific sequence is necessary to fulfil the task and the changes in the sequence lead to the abnormal behaviour of the robot. In our use case, each robot needs to be executed with a specific sequence of commands and change in the command conditions will disturb the entire plant. This is explained in detail in the following subsection with the example. All these types are injected into the data set in different patterns to make the data set ready for training the neural network for FDIA identification. Apart from generating the dataset with injecting attacks this MATLAB function also creates the labels for the dataset in parallel as normal and attack. These labels play a key role in training the neural network" [130].

"The GUI based attack injection tool is displayed in the Fig. 73. It is created to inject the attacks into the data during the run time to test the performance of the trained neural network. For the case

of testing, the GUI will inject attacks into a specific measurement values or command values of Robot 1 (ABB) and Robot 2 (KUKA). The type 1 and type 2 are of attacks on position measurements of robots with different amplitudes and the type 3 is attack on packet information by visual sensor and type 4 is attack on the control commands of the robot. The attacks injected in the testing phase varies from the attacks injected in the data set during the training phase. This makes efficient testing of our developed neural network for FDAI identification" [130].

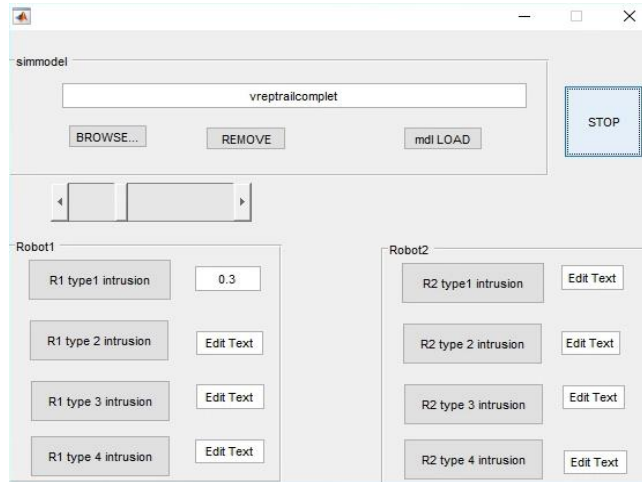


Figure 73. GUI based attack injection tool

Dataset development

"A discussion about the attack on control command sequence is necessary out of different types of attacks mentioned above. V-REP provides different ways to control the different joints of the robots. They are forward kinematic mode, inverse kinematic mode, dependent mode, motion mode and torque and force mode. The detail description of each and every mode is explained in [193]. Out of these modes, we choose forward kinematic mode to control our ABB and KUKA robots in our use case. In this mode, we can assign the specific movements of the joints to a certain position in relation to the environmental coordinates. These passive movements of the individual joint control define the functionality of the robot. Let us consider the ABB 6-axis robot used in the simulation environment shown in Fig. 74" [130].

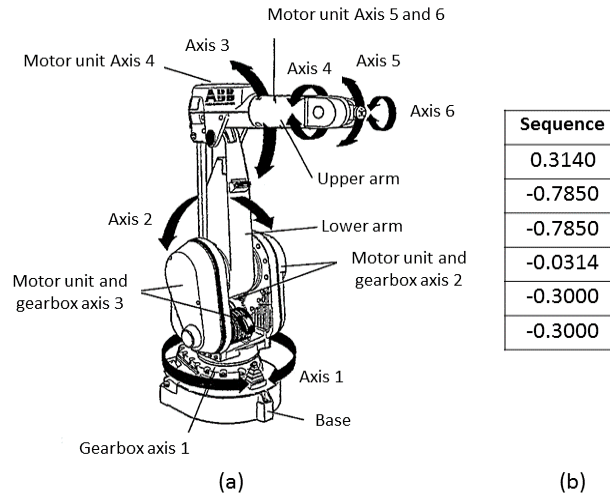


Figure 74. (a) 6-Axis ABB Robot [194] (b) Control command sequence

"The different axes of the robot are represented in Fig 74 (a). Based on the application we need to program the motion of the each and individual robot joint. In forward kinematic mode based robot control, we send the sequence of commands from MATLAB to the individual joints (axis) of the robot which makes the gripper to move to the specific position. The sequence of action commands is represented in Fig. 74(b). The command values in the sequence make the ABB robot to do the pick function. These sequences of action need to be continuously sent from MATLAB to V-REP simulation to run the simulation. After the sequence of commands were sent, we obtained the information back from V-REP to confirms the actuator action. In the same way, we also define the sequence of control commands for KUKA robots too. The switching of the control commands is based on the sensor information and the responses of the actors as a feedback from the V-REP environment" [130].

"After defining the sequence of commands, the robots will perform the defined functionalities which were explained earlier. Now, the simulation is initially run for 30 min and the obtained data is saved to the MATLAB workspace. Later these data are injected with FDIA using the MATLAB function explained in subsection IV.a. Different types of false data were injected into the sequence of commands" [130].

"As explained in above subsection, Fig. 75 (a) represents the original sequence and the attack injected sequence in grey color. The position information of the individual joints is replaced with the different values. Fig. 75 (b) represents changes in the task initiation. Here in this example, we

can see that the Task1 is replace by Task 2 due to the injection of attacks in the sensor information. Finally, Fig. 75 (c) represents the attacks which makes the change in the command sequences. Here all the measurements or commands will be in the normal range but the sequence of those command to be executed will be varied. This makes them more complex to identify and a proper training and sufficient amount of dataset is necessary to make them learn by our machine learning algorithms" [130].

"Based on the types of defined attacks the attack injection MATLAB function will insert the specified attacks on the dataset. As well as inserting the attacks in parallel the labels for the dataset is also created. The labels are necessary to train and test the performance of the developed ANN for FDIA identification. In total 20% of the data is injected with attacks in the dataset" [130].

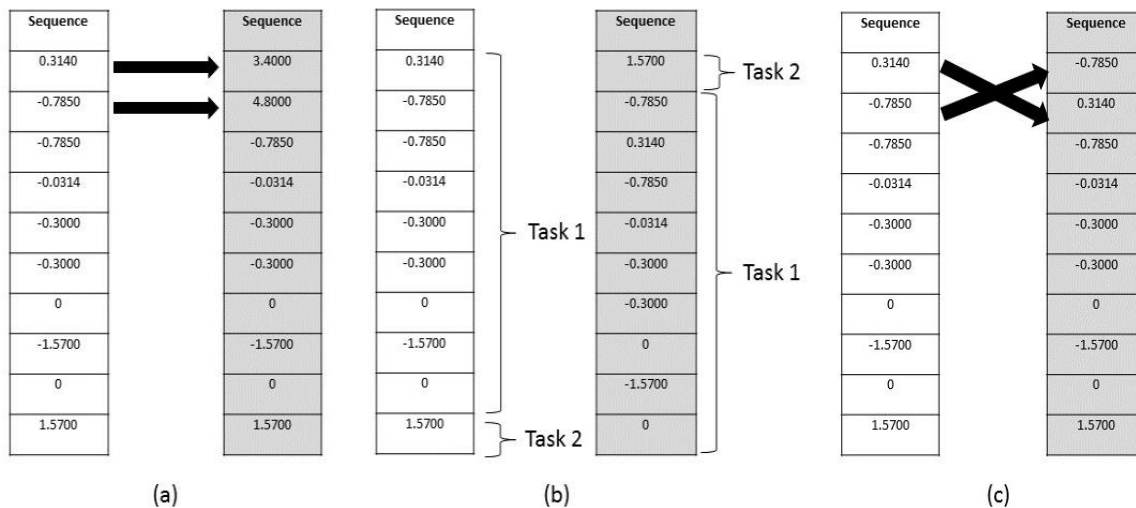


Figure 75. Attack on (a) Position Measurement (b) Sensor Measurement (c) Command sequence

Approach

"The workflow of our approach for identifying the FDIA is shown in Fig. 76. As explained earlier our main purpose is to detect abnormal behaviours caused by FDIA in plant data, for this purpose we created a manufacturing plant use case in Virtual Robotic Experimentation Platform (V-REP) to generate continuous data" [130].

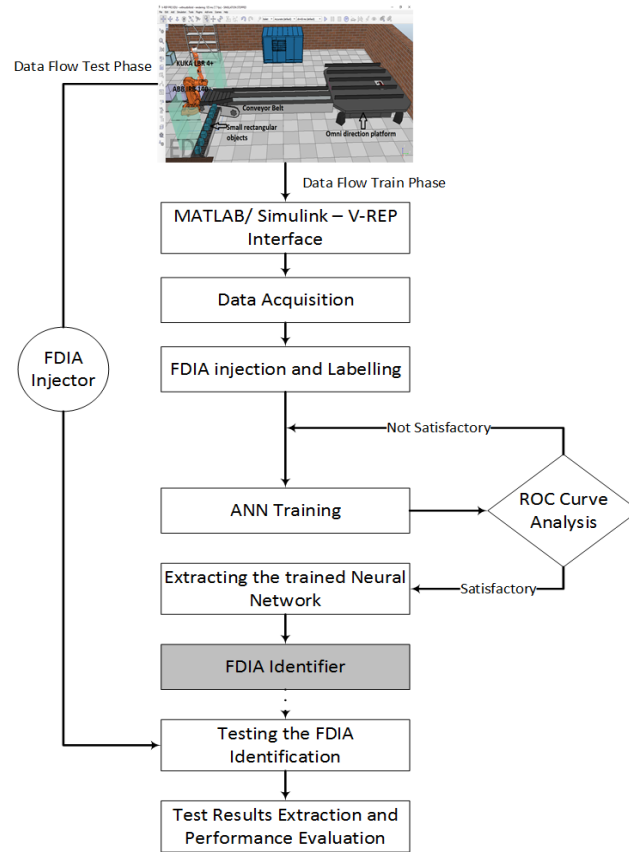


Figure 76: Workflow of ANN based FDIA identifier

"This data is initially used for training of our neural network for FDIA in the data and later for testing the performance. Fig. 76 shows two data paths from the plant. One is used for training purpose and the second one for testing purpose. V-REP is a simulation tool for manufacturing and assembly applications which provides interfaces to different remote API's. in the scope of this work A detailed description of the developed use case is discussed in below subsection. We use the V-REP interface to MATLAB. This interface is used to acquire the simulated data from the plant to MATLAB. Through the interface, we the acquired data initially saved in MATLAB workspace. The simulated data of the plant is attacks free. We developed an attack injection tool to inject attacks into the data before training the neural network. Different attacks were injected into the data and the data is labelled with the normal and attack data. This injected data along with labels is used for training the neural network. The Receiver Operating Characteristic (ROC) curve is analyzed to identify the performance of the trained NN. If the classification accuracies were satisfactory, we extract the trained neural network. This extracted neural network is considered as

an FDIA identifier and placed in the Simulink model for test purpose. If the ROC curve is not satisfactory, we perform further training of neural network by changing the parameters of ANN. During testing, new data from the plant is taken and attacks were injected during run time. The performance of the ANN based FDIA identifier were evaluated with different performance parameters. A detailed description of each and every step represented in Fig.1 is explained in the following sections" [130].

Shallow Neural Network for FDIA identification

"The following Fig.77 will give an overview on the training phase and testing phase of the neural network based FDIA identification" [130].

Training Phase

"From Fig. 77 (a) we see the training phase of our neural network based FDIA identification. The simulated use case delivers different signals of the plant to Simulink environment. The signals comprise of both the sensor and the actuator signals of the plant. The simulated plant was running for a specific period of time and MATLAB/Simulink acquires the periodic data samples of the plant which are initially stored in the environment workspace. The acquired data is injected with different types of attacks explained earlier. In order to run the plant continuously the sensor and actuator signals must be given to the control unit and the output signals are fed back to the plant" [130].

"For attack identification, we used ANN. ANN's can generalize and identify specific patterns in the dataset which are very complex to generate by an individual. They can classify the normal and attack patterns after proper training. The training is done in supervised manner" [130].

"It uses a non-linear regression to abstract information and differentiate normal from attack. We trained our neural network with the generated attack dataset and labels. The input of the neural network is the sequence at every instance of time. The ANN tries to learn the relation between the inputs and outputs. The hidden layer has 100 neurons to train efficiently for all possible attack identification. For the case of simplicity, we used in this experiment only one hidden layer" [130].

"The response of the neural network is reviewed and the configuration of the system is refined until the neural network analysis reaches a satisfactory level. Cross validation methods are used to

estimate an appropriate stopping point for gradient descent search and thus minimize the risk of overfitting" [130].

"Scaled Conjugate Gradient algorithm is used to train the feed forward neural network in MATLAB which gives high classification accuracy in pattern recognition applications. We constructed the neural network with one hidden layer and 100 hidden neurons in the hidden layer. We used sigmoid transfer function for better classification. Currently, the neural network is able to classify the input data into four classes. The normal values and the three attack types mentioned in Fig. 75. The training process is continued until the best validation performance is achieved" [130].

"This parameter indicates the point where the accuracy over the validation performance stays same or decreases. At this point the training of neural network is stopped to prevent overfitting of the neural network" [130].

"During the training process, we split the data into training, validation and testing phase. We choose 65% of data for training 15% for validation and 20% for testing. The performance of the training process is evaluated with the confusion matrix generated after reaching the satisfactory level of training process. Confusion matrix indicates the ability of the classifier. The generated confusion matrix shows the ability of the trained neural network at the best validation performance point which display the accuracies of normal as well as attack data" [130].

Test Phase

"Finally, the testing phase evaluates the trained neural network based. From Fig. 77 (b) we see the testing phase of our neural network for attack identification. During the testing phase, we get the live data from the plant continuously. This data is initially acquired by data acquisition which acts as an interface between MATLAB/Simulink and V-REP. The obtained data is given to train neural network for attack identification as well as to the control unit as shown in Fig. 77 (b). Different types of attacks are injected into different parts of the robot during run time and can be used to test the performance of the developed FDIA identification system. Whenever an attack is detected it raises an alarm and halts the simulation. If no attack is detected the control signals were given back to the simulation and the process continues" [130].

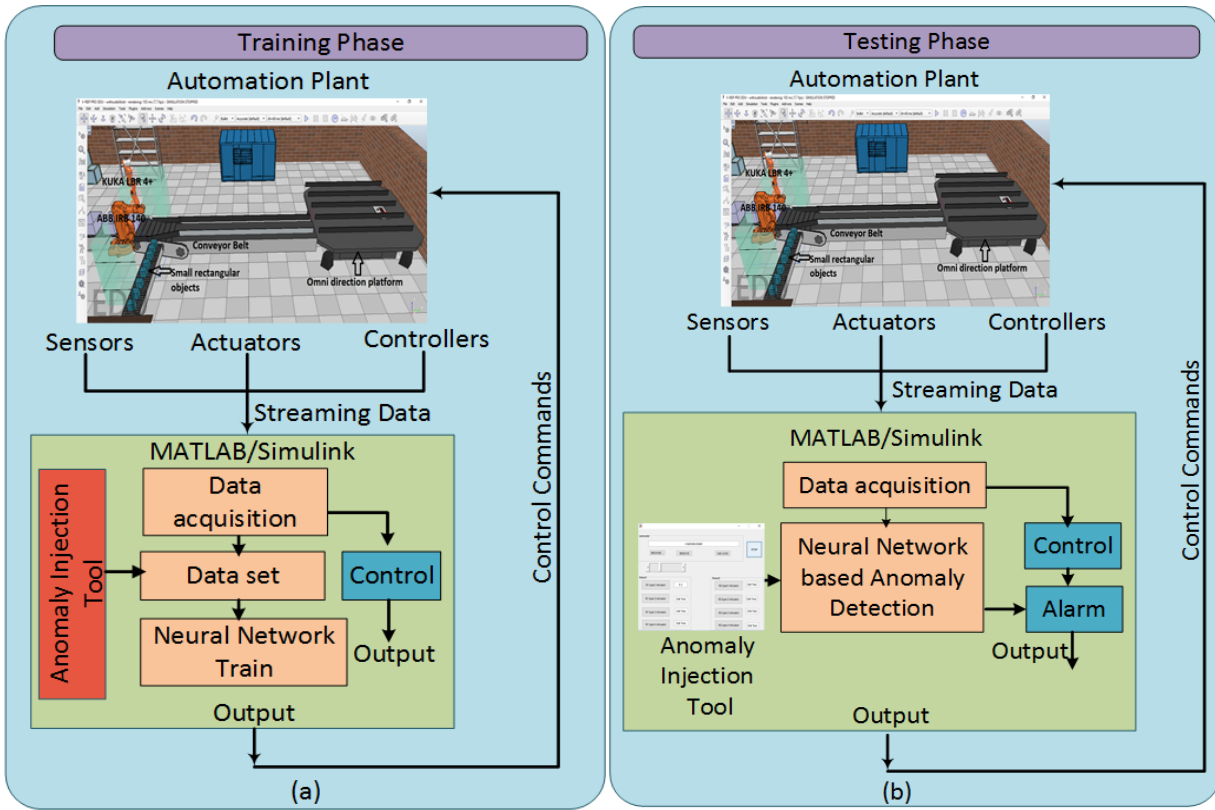


Figure 77: ANN based FDIA identification (a) Training phase (b) Testing phase

Results and Evaluation

"We evaluated the performance of our ANN based FDIA identification with the following attributes resulted from training and testing datasets generated from the manufacturing plant. We first identify the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) from generated confusion matrix then the performance parameters are evaluated. The definition of TP, TN, FP and FN are discussed below followed by the performance evaluation parameters Accuracy (ACC), Precision (P), Recall (R) and F-Measure (F)" [130].

"The evaluate performance metrics were shown in the following Table C-1. Apart from the above-mentioned parameters, the performance of the neural network also depends on the initial conditions, no. of hidden layers and no. of neurons in the hidden layers and the transfer function used by the neurons. We test the performance of the neural network with different mentioned conditions and the best results produced were as shown" [130].



Figure 78. Confusion matrix of the Neural Network at best validation performance

Appendix C

Table C-0-1: Evaluation of performance parameters for identification of injection attacks in automation applications

Parameter	Value
Accuracy	99.74
Precision	99.70
Recall	100
F- Measure	99.84

Appendix D

Simulation of network attacks dataset for future applications

Despite using NSL-KDD dataset as well as UNSW-NB 15 dataset for identifying network attacks the attacks are not update and every day we see a new type of attack being appeared. Data collected from an operational computer network is optimal for evaluation of IDS, but this data can contain personal or sensitive information. Due to this reason, as a part of this thesis, the effort of generating the own network attack dataset with different datasets is explored. These new datasets are necessary to evaluate the developed IDS as well as to make the developed IDS robust against the novel attacks. Hence, the network attack data was synthesized and recorded on a network which simulates an operational network of a common user connected to the internet.

Simulation network consisted of a Linux machine connected to multiple virtual machines running on the Linux operating system. The types of attack categories and different attach types we simulated were listed in the Table D-1 below:

Table D-0-1: Planned attack classes and their types

Attack Categories	Attack Types
Distributed / Denial of Service	Neptune, Ping of Death, Smurf, Land, UDP, Strom
Probe	Nmap, IPscan

Different tools we used for generating those attacks. They are:

Hping3: Command line oriented TCP/IP network packet assembler/analyzer, inspired by ping unix command [195].

Nmap: A Securirty scanner for discovering hosts and services on a computer network [196].

Cmd: A linux command prompt

Angry IP Scanner: It is a very fast IP address and port scanner in a network [197].

Attack classes and attack types

Denial of Service (DoS) and Distributed Denial of Service (DDoS)

A denial of service attack is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, therefore denying legitimate users access to a machine. Some common types of denial of service are Neptune and Ping of Death which mostly use the vulnerability of TCP Handshake.

TCP Handshake implementation:

When hosts need to establish communications via the TCP transport protocol, they must do a session initiation, which consists of a three-way handshake:

1. The source host initiated the communication by sending a TCP packet to the destination host the SYN flag (SYNchronize sequence numbers) set to 1. In this packet reside the source IP address and port number as well as the destination IP address and port numbers.
2. The destination host responds by sending a TCP packet to the source host with the flags SYN and ACK (ACKnowledge) set to 1. The response is sent to the source IP address and port of the initial packet in step 1.
3. The source host sends the destination host another TCP packet with the ACK flag set to 1. This completes the 3-way handshake and normal data communication can start.

Attack types in DoS and DDoS attack class are:

Neptune

A SYN flood DDoS attack exploits a known weakness in the TCP connection sequence (the “three-way handshake”), where in a SYN request to initiate a TCP connection with a host must be answered by a SYN-ACK response from that host, and then confirmed by an ACK response from the requester. In a SYN flood scenario, the requester sends multiple SYN requests, but either does not respond to the host’s SYN-ACK response or sends the SYN requests from a spoofed IP address.

Each half-open TCP connection made to a machine causes the server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections.

The half-open connections data structure on the victim server system will eventually fill and the system will be unable to accept any new incoming connections until the table is emptied out.

A Neptune attack can be distinguished from normal network traffic by looking for a number of simultaneous SYN packets destined for a particular machine that are coming from an unreachable host.

Attack Generation by using Hping3 tool

➔ `hping3 -S -flood -V (dest ip addr)`

To specify the source(80)/destination port(5050)

➔ `hping3 -V -S -p 80 -s (dest port)(dest ip addr)`

For DDOS flooding:

➔ `hping3 -S -flood -V -rand -source (dest ip addr)`

Ping of Death

Ping of Death is a type of network attack in which an attacker sends a network packet that is larger than what the target computer can handle. This can crash the computer or freeze or degrade computer service. Ping of death is used to make a computer system unstable by deliberately sending larger ping packets to the target system over an IPv4 network.

An attempted Ping of Death can be identified by noting the size of all ICMP packets and flagging those that are longer than 64000 bytes.

Attack Generation

➔ `ping -l 65510 your.host.ip.address`

Land Attack

The Land attack occurs when an attacker sends a spoofed SYN packet in which the source address is the same as the destination address.

The Land attack is recognizable because IP packets with identical source and destination addresses should never exist on a properly working network.

Attack Generation

➔ `hping3 -S (targetIP) -a (targetIP) -k -flood`

UDP Storm

“UDP flood” is a type of Denial of Service(DOS) attack in which the attacker overwhelms random ports on the targeted host with IP packets containing UDP datagrams.

The receiving host checks for applications associated with these datagrams and finding none sends back a “Destination Unreachable” packet. As more and more UDP packets are received and answered, the system becomes overwhelmed and unresponsive to other clients.

This type of attack can be distinguished from normal network traffic by looking for a number of simultaneous UDP packets destined for a particular machine that are coming from an unreachable host.

Attack Generation

`hping3 -2 -S -flood -V (dest ip addr)`

Probing

In recent years, a growing number of programs have been distributed that can automatically scan a network of computers to gather information or find known vulnerabilities. These network probes are quite useful to an attacker who is staging a future attack. An attacker with a map of which machines and services are available on a network and can use this information to look for weak points. Some of these scanning tools enable even a very unskilled attacker to very quickly check hundreds or thousands of machines on a network for known vulnerabilities.

Nmap

Nmap is a general-purpose tool for performing network scans.

This program also allows a user to specify which ports to scan, how much time to wait between each port, and whether the ports should be scanned sequentially or in a random order.

The signature of a port scan using the Nmap tool varies widely depending on the mode of operation selected. A port scan can be recognized by noting that network packets (whether via TCP or UDP,

or via only FIN packets or only SYN packets) that have been sent to several (or more) ports on a victim or group of victims within some window of time.

Probe:

→ Nmap (victim IP)

IPScan

An IP sweep attack is a surveillance sweep to determine which hosts are listening on a network. This information is useful to an attacker in staging attacks and searching for vulnerable machines.

An intrusion detection system looking for the simple IP sweep used in the simulation can look for many Ping packets, destined for every possible machine on a network, all coming from the same source.

Probing is done using Angry IP scanner

[Dataset generation](#)

A connection record summarizes the packets of a communication session between a connection initiator with a specified source IP address and a destination IP address over a pair of TCP/UDP ports.

A ready-to-use Linux OS is configured to route all traffic through the network. The linux machine is composed of 2 virtual machines for generating the hping attacks. The outgoing traffic is captured and pcap files are generated.

The labelled connection records in the training set are to be categorized normal or DOS/DDOS or Probe. The basic features are directly extracted or derived from the header information of IP packets and TCP/UDP segments in the packet capture files of each session.

The pcap files are extracted using tshark which is a companion tool to Wireshark [198].

TShark works from the command line rather than a graphical interface. TShark allows access to the raw bytes from a packet capture, it is possible to not only replicate, but also extend Wireshark's extraction functionality.

Command to extract pcap file:

→ tshark -I (interface) -w /tmp/(filename.pcap)

Feature Generation:

Measurement of network traffic is done at the IP flow level. IP flow level data is a unidirectional series of IP packets of a given protocol traveling between a source and a destination IP/port pair within a certain period of time.

For features generation, a new application, the ISCXFlowMeter [199] was used to generate the flows and calculate all necessary parameters. ISCXFlowmeter is a network traffic flow generator written in java and offers flexibility in choosing the features and adding new ones.

The FlowMeter generates bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence the statistical time-related features are also calculated separately in the forward and reverse direction.

The output of the application is the CSV file format with more than 80 features. The features available from the application are described in the table below:

Appendix D

Table D-0-2: Features of the generated dataset

Attribute No.	Attribute Name	Description	Sample data
1	ts	Times stamp of the first packet	1533886081.0808
2	id.ori_h	Source IP address	192.168.0.104
3	id.ori_p	Source port number	46062
4	id.resp_h	Destination IP address	88.221.214.42
5	id.resp_p	Destination port number	80
6	proto	Transport layer protocol of the connection	tcp
7	duration	Duration of the connection	0.008459
8	orig_bytes	Number of payload bytes the originator sent	1025
9	resp_bytes	Number of payload bytes the responder sent	1838
10	conn_state	Connection state description	SF
11	orig_pkts	Number of packets that the originator sent	2

Appendix D

12	orig_ip_bytes	Number of IP level bytes that the originator sent	104
13	resp_pkts	Number of packets that the responder sent	1
14	resp_ip_bytes	Number of IP level bytes that the responder sent	52