A Lightweight Supervised Intrusion Detection System for Future Smart Grid Metering Network

Sotiris Chatzimiltis

*Master of Science in*

*Computer Vision, Robotics and Machine Learning*

from the

University of Surrey



*Department of Electronic Engineering*

Faculty of Engineering and Physical Sciences

University of Surrey

Guildford, Surrey, GU2 7XH, UK

September 2022

Supervised by: Dr. Mohammad Shojafar

©Sotiris Chatzimiltis 2022

Declaration of originality

I confirm that the project dissertation I am submitting is entirely my own work and that any material used from other sources has been clearly identified and properly acknowledged and referenced. In submitting this final version of my report to the JISC anti-plagiarism software resource, I confirm that my work does not contravene the university regulations on plagiarism as described in the Student Handbook. In so doing I also acknowledge that I may be held to account for any particular instances of uncited work detected by the JISC anti-plagiarism software, or as may be found by the project examiner or project organizer. I also understand that if an allegation of plagiarism is upheld via an Academic Misconduct Hearing, then I may forfeit any credit for this module or a more severe penalty may be agreed.

A Lightweight Supervised Intrusion Detection System For Future Smart Grid Metering Network

Sotiris Chatzimiltis

Author Signature: Sotiris Chatzimiltis Date: 02/08/2022

Supervisor’s name: Dr Mohammad Shojafar

Word Count

Number of Pages: 53

Number of Words: 11152

Abstract

The integration of information and communication technologies into the power generation, transmission and distribution system provides a new concept called Smart Grid (SG). The wide variety of devices connected to the SG communication infrastructure generates heterogeneous data with different Quality of Service (QoS) requirements and communication technologies. Hence, this project aims to design a robust IDS dealing with anomaly SG data and impose the proper defence algorithm to alert the network. An intrusion Detection System (IDS) is a surveillance system monitoring the traffic flow over the network, seeking any abnormal behaviour to detect possible intrusions or attacks against the SG system.

In this project, we proposed an efficient and lightweight distributed IDS model using supervised detection machine learning algorithms and feature degradation techniques to boost the detection rate as well as decrease the energy and resource consumption of smart meters. As shown in the preliminary results section an accuracy of 83.45% was achieved when the initial dataset was used, and an accuracy of 81.98% when the PCA dimensionality reduction method was utilized. The main disadvantage is the high FNR when the initial dataset was used for training there was 26.50% FNR and when the PCA algorithm was used there was 28.09% FNR.

Future steps include the use of different feature selection algorithms and the combination of hyperparameter finetuning to achieve better overall performance. Finally, the final IDS model will be developed using horizontal federated learning where multiple smart meters (clients) will communicate with a server to create one model using distributing machine learning.

Table of Contents

Declaration of originality ii

Word Count iii

Abstract iv

Table of Contents v

List of Figures vii

List of Abbreviations ix

1 Introduction 1

1.1 Background and Context 1

1.2 Scope and Objectives 2

1.3 Achievements 2

1.4 Overview of Dissertation 2

2 Background theory and literature review 3

2.1 Electrical Grid 3

2.2 Smart Grid 3

2.3 Advance Metering Infrastructure 4

2.4 Intrusion Detection Systems 4

2.4.1 Signature-based IDS 5

2.4.2 Anomaly-based IDS 5

2.4.2.1 Statistical-based Techniques 6

2.4.2.2 Knowledge-based Techniques 6

2.4.2.3 Machine Learning-based Techniques 6

2.4.3 Stateful-Protocol-Analysis (SPA) 6

2.5 IDS for Smart Grids 7

2.6 Related Works 8

2.7 Summary 11

3 Methodology 12

3.1 General Smart Grid IDS architecture 12

3.2 Proposed Smart Meter IDS architecture 13

3.2.1 Spiking Neural Networks (SNN) 14

3.2.2 Traditional Machine Learning Techniques 16

3.2.2.1 Multi-Layer Perceptron (MLP) 16

3.2.2.2 Logistic Regression (LR) 17

3.2.2.3 Decision Trees (DT) 17

3.2.2.4 Random Forest (RF) 17

3.2.2.5 Gradient Boosting Classifier (GBC) 18

3.2.2.6 K-Nearest Neighbors (K-NN) 19

3.3 Federated Learning 20

4 Implementation and Results 22

4.1 Data Pre-processing 22

4.2 SNN Hyperparameters 23

4.3 Evaluation metrics 24

4.4 Binary Classification 24

4.5 Multiclass Classification 29

4.6 NAN Federated Learning 32

4.6.1 Binary Classification 33

4.6.2 2 Clients Experiment 37

5 Conclusion and Future work 39

References 40

Appendix 1 43

List of Figures

[Figure 2.1: Comparison of components between Smart Grid (left) and conventional Power Grid (right) [7] 4](#_Toc109987472)

[Figure 2.2: Intrusion detection framework proposed by [15] 7](#_Toc109987473)

[Figure 2.3: Flowchart of intrusion detection procedure from smart meter to AMI headend. [1] 9](#_Toc109987474)

[Figure 3.1: Overview of how IDS will be placed in the Smart Grid 12](#_Toc109987475)

[Figure 3.2: Components of IDS for a single Smart Meter 13](#_Toc109987476)

[Figure 3.3: Leaky integrate-and-fire neuron model [17], recurrent representation of a neuron (left), unrolled graph of the neuron (right). 14](#_Toc109987477)

[Figure 3.4: Illustration of rate coding for different pixel values. Black pixel has zero intensity thus no spikes appear, whereas white pixel has intensity of 1 thus spikes appear in every time step [17]. 15](#_Toc109987478)

[Figure 3.5: MLP with one hidden layer illustration 16](#_Toc109987479)

[Figure 3.6: Decision Tree example [18] 17](#_Toc109987480)

[Figure 3.7: Example of Random forest [18] 18](#_Toc109987481)

[Figure 3.8: Gradient Boosting Classifier concept illustration [20] 19](#_Toc109987482)

[Figure 3.9: K-NN simple example 19](#_Toc109987483)

[Figure 3.10: Horizontal Federated Learning workflow example 20](#_Toc109987484)

[Figure 4.1: Best dataset/ machine learning technique combination accuracies 26](#_Toc109987485)

[Figure 4.2: DT confusion matrices, initial dataset (left), PCA dataset (right) 27](#_Toc109987486)

[Figure 4.3: SNN confusion matrices, initial dataset (left), PCA dataset (right) 27](#_Toc109987487)

[Figure 4.4: MLP confusion matrices, initial dataset with standard scaler (left), initial dataset with minimax scaler (middle), PCA dataset (right) 28](file:///C:\Users\sotir\Desktop\Masters\Dissertation\Final_Report\MScIFinalDissertation_SotirisChatzimiltis_new.docx#_Toc109987488)

[Figure 4.5: Multiclass classification accuracies of the seven combinations 29](#_Toc109987489)

[Figure 4.6: Confusion matrix of Initial\_m and MLP combination 30](#_Toc109987490)

[Figure 4.7: Confusion matrix of Initial\_m and DT combination 30](#_Toc109987491)

[Figure 4.8: Confusion matrix of Initial\_m and SNN combination 31](#_Toc109987492)

[Figure 4.9: MLP Federated Learning Accuracy Chart 34](#_Toc109987493)

[Figure 4.10: LR Federated Learning Accuracy Chart 34](#_Toc109987494)

[Figure 4.11: SNN Federated Learning Accuracy Chart 35](#_Toc109987495)

[Figure 4.12: Accuracy Comparison of ML techniques trained with 3 clients 35](#_Toc109987496)

[Figure 4.13: Confusion Matrix of SNN trained with 3 clients after the 1st round of training 36](#_Toc109987497)

[Figure 4.14: Accuracy comparison of ML Techniques when trained with the 2 Clients dataset 37](#_Toc109987498)

[Figure 4.15: MLP Cross Validation accuracy of each Client 38](#_Toc109987499)

# List of Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Definition** |
| AD | Anomaly-based Detection |
| AMI | Advance Metering Infrastructure |
| ANN | Artificial Neural Network |
| CIA | Confidentiality Integrity Availability |
| DC | Data Concentrator |
| DR | Detection Rate |
| DT | Decision Trees |
| FAR | False Alarm Rate |
| FL | Federated Learning |
| FNR | False Negative Rate |
| FPR | False Positive Rate |
| GBC | Gradient Boosting Classifier |
| HAN | Home Area Network |
| HFL | Horizontal Federated Learning |
| HIDS | Host-based Intrusion Detection System |
| IDS | Intrusion Detection System |
| K-NN | K-Nearest Neighbours |
| lr | learning rate |
| LR | Logistic Regression |
| ML | Machine Learning |
| MLP | Multi-Layer Perceptron |
| NAN | Neighbourhood Area Network |
| NIDS | Network-based Intrusion Detection System |
| NN | Neural Network |
| PCA | Principal Component Analysis |
| PSO | Particle Swarm Optimisation |
| R2L | Root machine to Local user |
| RF | Random Forest |
| RNN | Recurrent Neural Network |
| SD | Signature-based Detection |
| SG | Smart Grid |
| SM | Smart Meter |
| SNN | Spiking Neural Network |
| SPA | Stateful Protocol Analysis |
| WAN | Wireless Area Network |

# Introduction

<1 or 2 paragraphs to introduce the topic>

***Here in the first part of your introduction you need to make an introduction to the topic with one or two paragraphs. State what the topic is, where it is used and what its purpose is. The reader needs to be in no doubt over the topic area you are working in (e.g. communication networks, wireless systems, radio, computer vision, signal processing etc.) and the type of work it involves (e.g. theoretical, experimental, programming or hardware).***

## Background and Context

A power grid is an interconnected network with the scope of delivering electricity from the utility to the user, in one-way communication. Electric energy is generated, transmitted, and distributed to our houses. However, in most recent years a new concept was introduced, called Smart Grid (SG). In today's digital world, it is now possible to transition to a faster, smarter electricity grid that can (a) provide better quality electricity with two-way communication (of electricity and information between utility and user), (b) balance power supply and demand in real-time, smoothing out power demand peaks, and (c) make consumers active participants in the generation and consumption of electricity.

Smart Grid (SG) is an enhanced version of the traditional power grid. To be able to make use of the SG, some adaptation to the current network needs to take place. The introduction of new devices and services, that allow us to communicate, monitor, and control electricity and information, demands new protocols and standards [1], creating a composite system that is vulnerable to different types of cyber-physical attacks. The problem is that not all attacks can be faced using traditional ways of security (encryption, authentication), and since new attacks continuously arise we need new ways of security. Articles have been published about various Intrusion Detection Systems (IDS) developed, that detect suspicious behaviour in the SG, either by using the traffic flow of the network combined with various machine learning algorithms [1] or by using electricity measurements [2].

Smart Meters (SMs) are a key component of SG, is the device that allows the user to have real-time information about his energy consumption and pricing. SMs are also most vulnerable to attacks from intruders trying to compromise the device for their own good, thus developing an IDS model for SMs is crucial. Since SMs are resource-constrained devices we need to develop a lightweight system that will be able to detect anomalies in a continuously changing environment, without being resource-consuming. Machine learning algorithms will help us achieve our objective since by using a classification algorithm we will be able to classify between normal and suspicious activities. In this project, we are going to evaluate different machine learning algorithms and datasets and try to create our IDS.

## Scope and Objectives

1. Objective 1: Design a new SM architecture including a lightweight IDS model to protect data-stream demands
2. Objective 2: Design an IDS algorithm dealing with Smart Meters data traffic
3. Objective 3: Implement proposed IDS-enabled ML models on real and synthetic smart meter datasets
4. Objective 4: Verify the provided solution in different SM case studies (for multiple buildings and do analysis of different IDS models through entropy design mechanism).

## Achievements

A double layer IDS for SG is proposed, that includes a lightweight IDS for SMs and an IDS deployed in NANs. Components of SM architecture are analysed, and binary classification models were developed using SNNs and different variations of the NSLKDD dataset. Preliminary results are shown and discuss in order to identify the strengths, weaknesses and possible solutions that will allow better performance of the IDS when utilising SNN.

## Overview of Dissertation

Chapter 2 gives a more detailed overview about some of the necessary concepts of SG and IDS, as well as showcasing some previous work made on this topic and explaining how this project can mitigate certain gaps that occur in previous projects. Chapter 3 describes the proposed architecture of the IDS. Moreover, in Chapter 4 some preliminary results of an initial IDS are shown. Finally in Chapter 5 future work is discussed.

# Background theory and literature review

## Electrical Grid

Electrical power is an important part of our lives, most of our daily tasks demand some kind of electricity. Electricity travels from its start point to our houses and buildings through the so-called electrical/power grid. As shown in Figure 2.1 electrical power is most of the time generated in power plants and it is then transmitted and distributed to every building using transmission towers and distribution lines respectively [3].

However, the traditional grid establishes a unidirectional relationship between utility and user where we only have the electricity flowing from end to end and merely controlled by conventional meters that just display the amount of energy consumed by each household, company, hospital, etc. As technology advances, we can create a new enhanced version of the power grid, the Smart Grid (SG).

## Smart Grid

The smart grid is the elevated version of power grid infrastructure. However, there is not a guideline underlining what components the SG consists of, so it is still a vision [4]. SG’s scope is to improve the efficiency, reliability, and safety of the grid using automated control and communication technologies [5]. Furthermore, Figure 2.1 shows that the SG fuses renewable and alternative energy sources into the infrastructure, something that is was not used in the conventional power grid.

For a kind of infrastructure like this to work, there is a need to incorporate several devices and services [6]. Using those components the SG establishes a bidirectional relationship between utility and user where both electricity and information are exchanged. This gives the freedom to the end-user to have an instant representation of his consuming behaviour as well as enabling him to control different home appliances and smart things remotely [1]. As mentioned before SGs include several advancements thus meaning there is a need for new protocols and standards to support the communication between them, leading to an emerged need to develop different SG security systems.

Diagram

Description automatically generated

Figure .: Comparison of components between Smart Grid (left) and conventional Power Grid (right) [7]

## Advance Metering Infrastructure

AMI is the basic component of the SG. According to [8] is the system responsible to measure, collect, transfer, and analyze power usage, as well as communicating with metering devices. It allows the user to have an active role in the system, where each user can contribute to reducing peak demands. The AMI architecture consists of three main elements, the Smart Meter (SM), the Data Concentrator (DC) Unit, and the Smart Grid Control Center (headend). SMs record the consuming behavior of a property, as well as communicate with the control center to exchange information. DCs collect data from a number of nearby SMs and forward them to the control center. Moreover DCs act as the communication interposer between SMs and the control center. Finally, the SG control center is responsible to store and analyze the collected data from millions of SMs.

## Intrusion Detection Systems

Understanding what an IDS is, we first need to understand what an Intrusion is, in general. An Intrusion is an attempt of compromising the confidentiality, integrity, and availability (CIA), or to bypass some security measurements of a computer or a network [9]. Processes that can monitor network traffic and further analyse it to extract signs of intrusions are called intrusion detection processes. Although many events can appear to be bad in the network, sometimes we have some exceptions where normal traffic is detected as suspicious, thus we need a system that can correctly distinguish intrusions from normal network data. Therefore, an IDS is either a hardware or software system that automates the intrusion detection process.

IDSs can be deployed on a Host-Based or Network-Based scale. Host-Based IDSs (HIDS) is installed in a sole node and monitor the host’s data such as network traffic and logs on the system. On the other hand Network-Based IDSs (NIDS) observe the network’s traffic where it usually analyses the protocols that have been used to detect malicious behaviour [10]. A Hybrid-Based IDS can be created as well, where the strengths of both HIDS and NIDS are combined, thus creating a double layer IDS.

Several different approaches can be taken to implement an IDS, but they can be categorized into three classes, Signature-based Detection (SD), Anomaly-based Detection (AD), and Stateful Protocol Analysis (SPA).

### Signature-based IDS

Signature-based Detection (SD) used to be the naïve solution for IDS. The main function of an SD-IDS is to use pattern matching to identify a known attack. The main concept is to create a database of intrusion signatures and compare the host’s activities with the already existing signatures. If a match is found then an alarm is triggered [11]. SD has a major advantage rather than other approaches since it almost has an excellent detection accuracy for previously seen intrusions. However, it has a major flaw as well, SD cannot identify attacks that never saw before making it not a good choice to identify new attacks. After the first time, the signature is appended into the database for future use. Nowadays due to the increasing rate the zero-day attacks appear they made SD less effective thus less favorable to be used in IDS.

### Anomaly-based IDS

Anomaly-based Detection (AD) came to the surface in order to overcome the problems of SD. AD does not require prior knowledge of intrusion signatures in order to be used to detect attacks [12] however, it lacks in determining the specific category of an attack e.g. Denial of Service (DoS), unauthorized access from a remote machine to local user (R2L). AD systems work based on one simple assumption, that any behaviour observed that notably deviates from the typical one is classified as an anomaly and therefore is considered to be an intrusion. There exist three types of anomaly detection approaches, statistical-based, knowledge-based, and machine learning-based. In general, all of them use similar architecture which comprises training and a test stage [13].

#### Statistical-based Techniques

Statistical Anomaly IDS (A-IDS) techniques create a distribution model for normal behaviour profile and then detect events that happen with low probability and identify them as possible intrusions. Those models can be implemented using univariate, multivariate, or time series models [11]. Statistical A-IDS have multiple advantages, firstly they do not require any prior knowledge, they have the ability to learn the expected behaviour through observations. In addition, they have a very precise detection rate of intrusions over long periods of time. However, there are some drawbacks of using this technique for example those models can be trained by intruders [13].

#### Knowledge-based Techniques

This specific technique requires the creation of a knowledge base that reflects the normal traffic profile. The knowledge base is often created by an expert in the system. Knowledge base techniques show good performance as they reduce false-positive alarms since the knowledge for all normal behaviours is known. However, due to the continuously evolving environment, the knowledge needs to be reformed regularly which is a time-consuming task [12].

#### Machine Learning-based Techniques

ML techniques are the most used for A-IDS at the moment. This technique extracts useful patterns from large amounts of data in order to classify or predict behaviour. ML models can either be trained in Supervised, Unsupervised, or Semi-Supervised fashion depending on whether the data are labelled or not. ML models have the advantage of being flexible and adapting when the data are evolving, as well as providing a good and robust model. However, their main disadvantage is the high resource consumption and time needed to train a model, thus we need to consider which models we can train depending on the specifications of the machine that will be deployed on [13].

### Stateful-Protocol-Analysis (SPA)

SPA is the third detection technique and is the process of comparing pre-set profiles of accepted definitions of normal protocol activity for each protocol state against observed events to identify deviations. The word “Stateful” in SPA means that the specific IDS is able to understand and track the state of the network, transport, and application layer protocols. The “protocol analysis” methods used in SPA are based on protocol standards and as those standards are revised, IDS using SPA should be updated too. SPA has two major drawbacks, the first one is that they are very resource-intensive due to the analysis performed. The second one is the inability to identify intrusions that do not violate the characteristics of acceptable protocol behaviour [14].

## IDS for Smart Grids

SG is the fusion of the power system and a communication network, so malicious activities can be detected in the communication network and/or the physical power system. Generally, there is a further distinction of the networks inside the communication network of the SG. The communication network is made of Home Area Networks (HAN), Neighbourhood Area Networks (NAN), and the Wide Area Network (WAN). Figure 2.2 shows a graphical representation of those networks and what each network is comprised of.

IDSs can be deployed in each of those networks. Starting with the HAN, which consists of a smart metering device and multiple smart appliances. The HAN IDS will track the traffic arriving and leaving the HAN and if suspicious activity is detected then an alarm will fire to notify the user and/or the NAN IDS which will act as a second layer of security. NANs comprise of HANs that are closed to each other in a geological way. The NAN IDS will collect inbound and outbound data from all the HANs and will identify any intrusions. Finally, the WAN will act as a third layer to detect any intrusions [15].

Diagram

Description automatically generated

Figure .: Intrusion detection framework proposed by [15]

## Related Works

This section presents a literature review of some machine learning-based IDS for SG.

Khan et al. [16] proposed a feature selection IDS for SG. They evaluated two components of several IDS architectures before conducting their. Firstly the feature selection algorithm and secondly the classifier, outlining the advantages and disadvantages that they have. After comparing the architectures, they designed their own. The architecture consisted of a dataset selection step, the pre-processing step, the optimal feature selection step using particle swarm optimisation (PSO), and the training and evaluation of different conventional machine learning (ML) classification algorithms (K-nearest neighbours (K-NN), Neural Network (NN), Decision Tree (DT) and Random Forest (RF)) . The datasets used for the experiments were KDD99 and NSLKDD, which are used in this project as well. They performed both binary and multiclass classification. Table 2.1 shows the accuracy, the detection rate (DR) and the false alarm rate (FAR) for binary classification between normal and anomaly traffic.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | KDD99 |  |  | NSLKDD |  |  |
| Model Name | Accuracy % | DR% | FAR% | Accuracy % | DR % | FAR % |
| PSO + K-NN | 99.3 | 99.7 | 2.40 | 99.51 | 99.17 | 0.17 |
| PSO + NN | 99.2 | 99.2 | 0.50 | 97.54 | 98.18 | 3.13 |
| PSO + DT | 99.5 | 99.6 | 0.80 | 99.64 | 99.41 | 0.14 |
| PSO + RF | 99.6 | 99.6 | 0.60 | 99.65 | 99.3 | 0.08 |

Table .: Binary classification accuracy, DR and FAR comparison table of Khan et al. IDS

After creating the confusion matrix performance metrics are calculated as follows. Equation 1 shows that accuracy is the proportion of instances classified correctly as either normal or attack divided by the total number of instances in the dataset.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. |

Equation 2 shows that DR is the percentage of malicious traffic detected correctly, while Equation 3 calculates FAR. FAR divides the number of instances that are misclassified as attacks over the total number of instances that are classified as attacks. The optimal goal of every IDS is to have as high DR as possible and as low FAR as possible.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. |
|  |  | Eq. |

It is clear that the best performance comes when using the proposed pre-processing method and random forest. They report an accuracy of 99.65% , a detection rate of 99.3% and a false alarm rate of 0.08% which are pretty impressive results.

Faisal et al. [1] design an IDS architecture that takes place in the whole AMI. Their idea was if there is an intrusion detected in a HAN (i.e. in a SM IDS) or a NAN (i.e. in a DC IDS) trigger an alarm so networks that reside above (headend IDS) can check again if there is indeed an intrusion or not. Figure 2.3 shows the flowchart of the IDS architecture.

![Diagram

Description automatically generated]()

Figure .: Flowchart of intrusion detection procedure from smart meter to AMI headend. [1]

Furthermore, this paper gave emphasis on utilising evolving algorithms. Evolving classification algorithms take into account the change of the data distribution over time (i.e. concept drift). Their idea was good since network data traffic from KDD99 and NSLKDD datasets can be considered as a data-stream. Experiments were performed using seven different classification algorithms and utilising the KDD99 and NSLKDD datasets as well. However, the paper does not mentioned if any pre-processing techniques took place before the training of the algorithms.

The performance of each algorithm was evaluated using multiple different performance metrics, such as accuracy, model size, running time and false positive and negative rate. Table 2.2 shows the performances of the classifiers train and tested using the NSLKDD dataset.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | Accuracy (%) | Kappa Statistics (%) | Model Size (KB) | Running Time (secs.) | FPR(%) | FNR(%) |
| Accuracy Updated Ensemble | 93.39 | 90.23 | 20632.44 | 3.96 | 3.26 | 8.70 |
| Active Classifier | 89.26 | 84.19 | 420.43 | 0.28 | 4.04 | 15.29 |
| Leveraging Bag | 95.65 | 93.60 | 34375.00 | 1.62 | 2.03 | 6.85 |
| Limited Attribute Classifier | 96.59 | 95.02 | 24596.75 | 92.47 | 2.49 | 3.39 |
| Bagging using ADWIN | 96.05 | 94.21 | 10212.59 | 3.35 | 2.15 | 5.55 |
| Bagging using Adaptive Size Hoeffding Tree | 95.60 | 93.58 | 7724.45 | 3.16 | 3.17 | 3.92 |
| Single Classifier Drift | 93.97 | 91.09 | 1611.43 | 0.30 | 2.49 | 8.23 |

Table .: Performances of the classifiers train and tested on NSLKDD dataset [1]

As we can observe, accuracies of the classifiers are neither bad nor really impressive, but they are in a good state. However, we can see that false positive rate (FPR) and false negative rate (FNR) are quite high for some classifiers. For example the Active Classifier has the worst performance with 89.26% accuracy and a FPR of 4.04% and an FNR of 15.29%, meaning that for every one hundred instances fifteen of them are incorrectly classified as normal while they are attacks, and four of them are incorrectly classified as attacks while they are normal. Furthermore we can see that the Limited attribute classifier which has the best performance needs the most memory (~ 24MB), which in a recourse constraint environment it may be forbiting.

Finally this paper discuss the feasibility of using those data stream classifiers in each of AMI’s IDSs. Starting with the SM-IDS, the algorithm to be chosen has to meet the memory requirement of the smart meter. Thus, leading in the option of choosing either the Active Classifier or the Single Classifier Drift. Although, active classifier has the best memory and time performance, it has poor performance in the other metrics. As another option single classifier drift shows an average performance concerning accuracy and detection rates, and has a respectable memory and time requirement. Moving on to the data concentrator, the memory constraint is not really a problem, however we need to take into account that data flow is more frequent. Thus a classifier such as Leveraging Bag can be used since it has a good performance and a low running time. Finally the IDS that is going to be deployed in the SG control center (headend) has to be able to process a huge number of instances in a limited amount of time since the data are flowing in a high rate. Active classifier and single classifier drift are the only options because they require under half a second running time.

<append the comparison here>

## Summary

Chapter 2 gives a brief background theory about what a SG is and its main components. Also various types of IDS are discussed, and how those IDS can be applied to a SG. Furthermore two papers are analysed outlining their approach to the problem. Khan et al. [16] gave more emphasis on which feature selection/ dimensionality reduction algorithm is more robust when developing IDSs with traditional ML. On the other hand, Faisal et al. [1], takes into account the constraints that each layer has, and so the selection of the ideal IDS differs at each level. <Two/three more papers>

# Methodology

## General Smart Grid IDS architecture

One of the main features of Smart Grids is to introduce a bidirectional relationship between user and utility. Figure 3.1 illustrates an abstract of the Smart Grid architecture. Electricity and information flows from the utility side to the user, and the user shares back to the utility information. IDS are trying to protect the communication channel between utility and the user.

Starting from the bottom up, at the home are network level (green region), where each household has its own smart meter. Smart meters, gather information concerning the electricity consumption, as well as communicating with other smart appliances installed at home. The user can login the SM to view these information, however an intruder can do the same. Introducing an IDS in that first level, we apply a first layer of defence against attacks.

Another layer of defence can be applied in the neighbourhood area network (blue region). Each NAN has a data concentrator where information from nearby SMs is collected. Thus, an IDS can be deployed in this level in order to analyse the information and detect intrusions that may pass the SM IDS.

Finally a third layer of defence can be used in the utility side (smart grid control center, red region), but for now this is out of the scope of this project.

A screenshot of a computer

Description automatically generated with low confidence

Figure .: Overview of how IDS will be placed in the Smart Grid

## Proposed Smart Meter IDS architecture

Figure 3.2, shows the components of the IDS of a smart meter. Initially, data traffic is captured, and passed through the pre-processing stage. Firstly data traffic is normalized using either the min-max or the standard normalization technique shown in Eq. 4 and Eq. 5 respectively. The second pre-processing step is to encode non-numeric values, one-hot encoding method was used. One-hot encoding creates a new column for every possible value of every feature. The final pre-processing step is applying either a feature selection or a dimensionality reduction algorithm, so that the final data are as compact and as descriptive as possible.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. |
|  |  | Eq. |

The next stage is to classify if the data traffic captured is either normal or malicious. There are multiple machine learning classifiers to be used in this stage but for this dissertation, seven different classifiers were chosen to be used (SNN, MLP, LR, DT, RF, GBC, K-NN). The last stage is to evaluate the classification prediction performance, since having a good classifier is an important part.

Graphical user interface

Description automatically generated with medium confidence

Figure .: Components of IDS for a single Smart Meter

### Spiking Neural Networks (SNN)

SNNs are a kind of artificial neural networks, with the difference being that rather than having the traditional artificial neuron (e.g. McCulloch-Pits) they trade it for a spiking neuron. Spiking neurons, produce a weighted sum of inputs but instead of forwarding the result into an activation function (e.g. sigmoid, ReLU), this sum contributes to the membrane potential *U(t)* of the neuron. The main condition is when *U(t)* passes a pre-defined threshold the neuron will emit a spike to successive connections [17].

|  |  |  |
| --- | --- | --- |
|  |  | **Eq. 6** |
|  |  | **Eq. 7** |

Equation 5 shows how to compute the membrane potential of a spiking neuron. Firstly, there is the decay rate of the membrane potential. The second part is the weighted sum of the input which is the same as conventional neural networks. Finally there is the reset part, equation 6 shows the possible values of *Sout[t], i*f *U[t-1]*  exceeded the threshold meaning the spiking neuron fired, then the spiking neuron should be reset by subtracting the threshold from the membrane potential at time *t*.

![Diagram

Description automatically generated]()

Figure .: Leaky integrate-and-fire neuron model [17], recurrent representation of a neuron (left), unrolled graph of the neuron (right).

Figure 3.3 illustrates the architecture of a single spiking neuron. From the left image we can extract how equation 5 emerges, implicit recurrence is the decay part and *V* (explicit recurrence) is the multiplication of *Sout[t]* and *-θ*. The right image shows an unrolled iteration of how the neuron operates.

Another brain inspired modification that SNNs have over the ANNs is the nature of the inputs to the network. SNN inputs are encoded to spikes, there are several types of encoding but rate coding is used. Rate coding concept is to convert the input intensity into a firing rate. An example from [17] follows.

Imagine having a sample image from the MNIST dataset **I** of size 28 x 28 and there is the need of converting intensity array values, to a 3-D tensor of size 28 x 28 x *t*, where *t* is the number of time steps. Each pixel is encoded independently, the normalized pixel value [0,1] denotes the probability of a spike appearing at any given time step. Finally using a Bernoulli distribution we identify at what time steps there will be a spike. Figure 3.4 shows 3 pixel values, where black has a normalized value of 0 , meaning there is zero probability of a spike over the time steps. The next, is a grey pixel that has a normalized value of 0.5 , meaning there is 0.5 probability of a spike appearing in each time step, so if we have 8 time steps , we are going to have 4 spikes. Finally, if the pixel is white, having a normalized value of 1 , spikes will appear in every time step.

![A picture containing diagram

Description automatically generated]()

Figure .: Illustration of rate coding for different pixel values. Black pixel has zero intensity thus no spikes appear, whereas white pixel has intensity of 1 thus spikes appear in every time step [17].

As with conventional ANNs, SNNs can be altered in order to approach various problems, one of them is an IDS model for a SM. Network Traffic flows from and to a SM, so by capturing and processing those data in a meaningful way we can train an SNN in order to classify if a specific traffic instance is malicious or not.

The architecture of an SNN allows it to be used in recourse constrained devices. Computing the output of a normal feed forward NN needs a vast amount of multiplications to be made which consume resources, whereas in spike-based approaches the calculation of the output is much simpler because mainly it is the addition of weights since spikes have values of either 1 or 0 [17].

### Traditional Machine Learning Techniques

As mentioned in section 3.2.1 most of the machine learning methods developed can be altered to approach any problem, this implies the below six more traditional models. Some models are tree-based so the data are just fitted, meaning we don’t have weights to adjust, on the other hand some techniques such as MLP and LR need to be trained in order to perform well. Techniques that involved training can be implemented with federated learning as well, whereas methods such as DT or RF can only be used as a stand-alone IDS.

#### Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron is a class of feed-forward neural networks. MLP was designed in order to solve problems that were not linearly separable. They consist of multiple McCulloch and Pitts neurons which form layer feed-forward networks as shown in Figure 3.5. An MLP neural network consists of an input layer, at least one hidden layer and an output layer. Hidden and output layers are active, whereas the input layer is inactive since it only forwards the data to the network. Each layer has one or more neurons and an independent neuron unit, also known as ‘bias’, which has a constant input value of one (1). The role of the bias unit is to help the network adapt more effectively to the provided data.

Diagram, engineering drawing

Description automatically generated

Figure .: MLP with one hidden layer illustration

#### Logistic Regression (LR)

Logistic Regression can be regarded as the simplest form of a NN. All the input features of an instance are passed through the logistic sigmoid function to produce a probability value. Depending on the probability value(output) the instance is then classified into the respective class. For example in binary logistic regression, if output probability is greater than 0.5 then the instance belongs to class one else in class zero.

#### Decision Trees (DT)

Decision Trees (DTs) are a supervised learning method used for classification and/or regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Figure 3.6 shows a simple example of how DTs work. It can be observed that the decision rules split the observations in a way that the resulting groups are as different from each other as possible but observations in the same group are as similar as possible .

Chart, diagram

Description automatically generated

Figure .: Decision Tree example [18]

#### Random Forest (RF)

Random forest is a classification method that combines the predictions produced from multiple decision trees. Each individual tree outputs a class prediction and the class with the most votes becomes the models prediction (shown in Figure 3.7). The performance of RF is based on just a simple concept. Having a large number of relatively uncorrelated models operating as a group, will outperform any of the individual models. In order to ensure that every decision tree inside a random forest is diverse as possible with respect to the others two methods can be used. The first one is bagging, where each individual tree takes a random sample from the dataset to be trained with instead of the whole dataset. The second method is feature randomness, that restricts the number of features that can be used to split a node in each decision tree, by selecting a random subset of the available features [18].

A picture containing diagram

Description automatically generated

Figure .: Example of Random forest [18]

#### Gradient Boosting Classifier (GBC)

As RF so GBC is a type of ensemble learning, more precisely called boosting. Boosting methods combine many weak predictors to form a single strong model. In Gradient boosting instead of using the data to fit a predictor at each iteration, the predictor is fitted to the errors made by the previous predictor [19]. An illustration is shown in Figure 3.8.

Diagram

Description automatically generated

Figure .: Gradient Boosting Classifier concept illustration [20]

#### K-Nearest Neighbors (K-NN)

K nearest neighbors is one of the simplest supervised classification technique widely used. K-NN method finds the closest K neighbors of an instance and assigns the most common class to that instance that the K neighbors have. A simple example is shown in Figure 3.9 where the class if the blue point if we consider the 3 nearest neighbors is star.

Chart

Description automatically generated with low confidence

Figure .: K-NN simple example

## Federated Learning

The creation of machine learning models demands training the various models using data. Gathering these data can be a difficult task. To be more specific in order to train an IDS that is going to be deployed in a SM there is the need of collecting huge amounts of data traffic from various SMs located in different physical locations. However due to the arising user privacy and data security awareness new ways have to be paved in order to achieve creating an IDS that the data to be used in training are stored as “data islands” in many physical locations [21].

Federated Learning is a concept introduced by Google in 2016 with scope of training a high-quality centralized model using data that were distributed across multiple clients [22]. Yang et al. give a detailed definition of federated learning [23].

For this dissertation the concept of HFL (horizontal federated learning) will be utilised, where devices have the same feature space but different space samples [23].

![Diagram

Description automatically generated]()

Figure .: Horizontal Federated Learning workflow example

Figure 3.10 illustrates HFL architecture and steps. Each device (client) has its own data, and therefore at the start point each device trains its own model. The weights of every model are encrypted and transmitted to the server where they are combined. The server then sends an aggregated new model to every device. Finally devices update their model with the new weights.

There are multiple ways that the server aggregates the weights. For this dissertation the FedAVG technique was used which is the most usual technique. FedAVG calculates the average of the weights.

HFL can be applied in neighbourhoods of SMs. Most of the smart meters produce data in the same feature space. Finally there is no exchange of data from the SMs which is another advantage since a main concern is data privacy and protection..

<say why we using FL, in order to create a NAN IDS>

## Summary

Firstly, in Chapter 3 a general SG architecture was introduced. Secondly a SM IDS architecture was proposed and a brief analysis of the different components of the IDS was made. Furthermore all the machine learning techniques used for this dissertation were analysed. Finally the concept of federated learning was introduced in order to show how a NAN IDS can be developed.

<summary of chapter , every chapter 1,2,3,4>

1 (what is going to be in the project)

The core of achievements

# Implementation and Results

In chapter 4 an extensive analysis of different experiments is made. The code for re-creating the experiments can be found at https://github.com/sotirischatzimiltis/MscThesis. The code for training and testing the SNN was based on snnTorch tutorials [17], the rest of the machine learning models were implemented using Scikit-learn [24], and the framework used for federated learning was Flower [25]. The dataset utilised for the experiments was the NSLKDD [26] which is vastly used as a benchmark ML dataset. The NSLKDD Test+ dataset was used for testing, Tavallaee et al. [26] and Su et al. [27] reported accuracies ranging from 72% up to 84.15%.

## Data Pre-processing

The NSLKDD dataset consists of a training and a testing set. The training set have 125973 instances and the test set has 22544. All instances have 41 features, 1 label denoting whether or not that instance is an attack, and 1 number denoting the difficulty of classifying that record. From the 41 features, 38 of them are numeric and 3 of them have non-numeric values.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Normal | Attack | Total |
| Train Data | 67343 | 58630 | 125973 |
| Test Data | 9711 | 12833 | 22544 |

Table .: Train and Test datasets record break-down for binary classification

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Normal | DoS | Probe | R2L | U2R | Total |
| Train Data | 67343 | 45927 | 11656 | 995 | 52 | 125973 |
| Test Data | 9711 | 7460 | 2885 | 2421 | 67 | 22544 |

Table .: Train and Test datasets record break-down for multiclass classification

Tables 4.1 and 4.2 show how many records of each category exist for binary and multiclass classification.

As mentioned in Section 3.2, all numerical features were normalised using either the min-max or the standard normalization method (creating two datasets each time), and all the categorical features were encoded using one-hot encoding, resulting in a dataset having a total of 123 features. For the purpose of the experiments we are going to refer to this dataset as the initial dataset .

The initial dataset was the altered by using either a feature selection or a dimensionality reduction algorithm. To be more precise 3 more smaller datasets were created. The first dimensionality reduction method applied was PCA, where a set of correlated features is transformed into a new smaller set of uncorrelated features (i.e. principal components). The main idea is to make the dataset as compact as possible and as descriptive as possible. The new dataset (called PCA dataset for this experiment) consists of 29 features when Minmax scaling was applied, and 89 features when Standard scaling was applied [28].

The third and fourth datasets were generated using random forest. RF can calculate the importance of every feature in the dataset. By setting a threshold of the importance we can keep features with importance more than the threshold while discarding the rest. The third dataset, (i.e. RF05) has features of importance of more than 0.05, thus 8 features only make up every instance. The final dataset (i.e. RF005) has features with importance of more than 0.005, there are 28 features for every record of the dataset [28].

## SNN Hyperparameters

SNNs have multiple tuneable hyperparameters. Starting with the most important one, the number of time steps. Number of time steps controls how inputs are represented to the spiking neurons. Since inputs are transform to represented as a probability, a naïve approach is to use a high number of time steps in order to have a better representation. For example by setting the time step number to 10 , a feature with probability of 0.58 will sometimes have as input 5 spikes and sometimes 6 spikes. On the other hand having a time step of 100, the same feature will have 58 spikes every time. Finding the best number of time steps depends on the problem and the dataset.

The second most important parameters is the decay (Eq.6). SNNs are considered a type of RNN where membrane potential at the previous time step is used in the computation of membrane potential at the current time step. Having a high decay number, means that membrane potential of previous time step will affect more the result of the current potential than when using a small decay. Again, decay is problem specific and also depends on the dataset as well.

Other tuneable parameters include the learning rate (lr), and the number of training epochs, and the general architecture of the NN (i.e. number of layers and hidden nodes).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Dataset | Time Steps | Batch Size | Decay | Epochs | Learning Rate | Hidden Layers | Hidden Nodes |
| Initial | 25 | 64 | 0.9 | 5 | 0.0005 | 1 | 3000 |
| PCA | 25 | 64 | 0.9 | 5 | 0.0005 | 1 | 3000 |
| RF005 | 25 | 64 | 0.9 | 10 | 0.0001 | 1 | 3000 |
| RF05 | 25 | 64 | 0.9 | 10 | 0.0001 | 1 | 4000 |  |

Table .: Hyper-parameters of SNN for every dataset

Table 4.3 shows the hyperparameters used to train each SNN depending on the dataset used. The final hyper-parameters were chose after performing a number of executions and each time only one hyperparameter was altered.

## Evaluation metrics

Experiments were performed for every dataset, using all seven of the machine learning models mentioned in Section 3. Models were evaluated using seven different metrics. Table 4.3 gives a brief explanation for every metric used.

|  |  |  |
| --- | --- | --- |
| Metric | Equation | Explanation |
| **Accuracy** | *(TP +TN) / (TP +FP +TN + FN)* | *The percentage of correctly classified instances, either as normal or attack* |
| **Precision** | *(TP /(TP+FP))* | *The percentage of positive class predictions, that are indeed positive* |
| **Recall / Detection Rate (DR)** | *(TP/(TP+FN))* | *The percentage of how many of the positive examples in the dataset were predicted correctly* |
| **F1-score** | *(2\*Precision\*Recall)/(Precision +Recall)* | *A single score the balance Precision and Recall together* |
| **FAR** | *(FP / (FP +TN))* | *The percentage of instances misclassified as attacks.* |
| **FPR** | *(FP /(FP +TP))* | *The percentage of instances wrongly categorized as attacks* |
| **FNR** | *(FN /(FN+TN))* | *The percentage of instances wrongly categorized as normal* |

Table .: Explanation of metrics utilised to evaluate the ML models

## Binary Classification

The first experiment undertaken is the binary classification of data traffic as either normal or malicious (attack). This experiment has two main purposes. Firstly to identify which datasets are most suitable (shown in Table 4.5 and mentioned in Section 4.1) and secondly to assess the performance of a standalone IDS that was implemented with every technique mentioned in Sections 3.2.1 and 3.2.2.

|  |  |
| --- | --- |
| *Dataset* | *Description* |
| *Initial\_m* | All the extracted features were used (123) and they were normalized using the MinMax Scaler |
| *Initial\_s* | All the extracted features were used (123) and they were normalized using the Standard Scaler |
| *PCA\_m* | The extracted features were passed through the PCA dimensionality reduction algorithm and the new features were normalized using MinMax Scaler |
| *PCA\_s* | The extracted features were passed through the PCA dimensionality reduction algorithm and the new features were normalized using Standard Scaler |
| *RF0.05\_m* | The extracted features were passed through a Random Forest feature selection algorithm and the final features (importance > 0.05) were normalized using Minmax Scaler |
| *RF0.05\_s* | The extracted features were passed through a Random Forest feature selection algorithm and the final features (importance > 0.05) were normalized using Standard Scaler |
| *RF0.005\_m* | The extracted features were passed through a Random Forest feature selection algorithm and the final features (importance > 0.005) were normalized using Minmax Scaler |
| *RF0.005\_s* | The extracted features were passed through a Random Forest feature selection algorithm and the final features (importance > 0.005) were normalized using Standard Scaler |

Table .: Description of Datasets used for binary classification

For every combination of dataset and machine learning technique, 5 executions were made and the best results were reported. Contrary to SNN that preliminary experiments were made to finetune the hyperparameters (Section 4.2), for the rest of the techniques the GridSearchCV method was used to find the best combination of parameters from a given set that they were then used to do the final executions.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset/ML** | **DT** | **RF** | **MLP** | **GBC** | **KNN** | **SNN** | **LR** | **Average** |
| Initial\_m | **82.29** | 77.92 | **84.31** | 81.21 | 77.1 | **83.45** | 77.1 | **80.48** |
| Initial\_s | 79.93 | 78.3 | **84.59** | 81.06 | 78.08 | 80.1 | 75.11 | **79.60** |
| PCA\_m | **81.98** | 78.94 | **84.45** | 80.66 | 77.31 | **81.98** | 74.13 | **79.92** |
| PCA\_s | 80.18 | 78.51 | 81.37 | 79.25 | 78.45 | 78.01 | 73.38 | **78.45** |
| RF0.05\_m | 73.69 | 73.44 | 73.82 | 74.89 | 74.05 | 75.23 | 74.56 | **74.24** |
| RF0.05\_s | 78.95 | 75.47 | 77.2 | 77.02 | 77.69 | 75.6 | 72.19 | **76.30** |
| RF0.005\_m | 78.88 | 74.29 | 79.58 | 78.08 | 75.05 | 78.64 | 71.31 | **76.55** |
| RF0.005\_s | 80.53 | 75.86 | 77.2 | 80.19 | 75.6 | 74.84 | 71.48 | **76.53** |
| **Average** | **79.55** | **76.59** | **80.32** | **79.05** | **76.67** | **78.48** | **73.66** |  |

Table .: Accuracies of every dataset/machine learning technique combination

Table 4.6 shows the accuracies for every combination of dataset and machine learning technique. It can be clearly seen that the best-performing dataset on average was the initial dataset (Initial\_m, having 123 features) and the features were normalized using the Minmax Scaler having 80.48% average accuracy, whereas the worst performing dataset was when the random forest feature selection algorithm was used with 0.05 importance threshold and the features were then normalized using the Minmax Scaler (RF0.05\_m, having 8 features) having 74.24% average accuracy.

Concerning the machine learning techniques, the best performing model on average, was multi-layer perceptron (MLP) with an average accuracy of 80.32% and the worst performing technique was logistic regression (LR) with an average accuracy of 73.66%. The best single combination was when multi-layer perceptron (MLP) and Initial\_s dataset was used with an accuracy of 84.59%. On the other hand the worst combination is when LR technique and RF0.005\_m dataset were used having an accuracy of 71.31%.

Figure 4.1 illustrates the best dataset and machine learning technique combinations. It can be clearly observed that MLP outperforms every other technique no matter the dataset. Comparing now how the features were normalized, in both cases when the initial and the PCA datasets were normalized using the Minmax scaler they outperform their respective dataset that the features were normalized using the Standard scaler.

Figure .: Best dataset/ machine learning technique combination accuracies

The best seven combinations are further analysed in order to observed and extract useful information. The combinations are shown in Table 4.7.

|  |  |
| --- | --- |
| Dataset/ ML | Accuracy (%) |
| Initial\_s & MLP | 84.59 |
| PCA\_m & MLP | 84.45 |
| Initial\_m & MLP | 84.31 |
| Initial\_m & SNN | 83.45 |
| Initial\_m & DT | 82.29 |
| PCA\_m & DT | 81.98 |
| PCA\_m & SNN | 81.98 |

Table .: Best Combinations

Figures 4.2, 4.3, and 4.4 illustrate the confusion matrices of the best seven combinations. A pattern can be easily extracted from those figures. Normal data are classified as normal data most of the times, whereas there is not a good classification rate of attack type data. A reason why is that some attack type data have similar feature values as normal type data and thus correctly classifying them becomes more difficult.

Chart, waterfall chart, treemap chart

Description automatically generated

Figure .: DT confusion matrices, initial dataset (left), PCA dataset (right)

Chart, treemap chart

Description automatically generated

Figure .: SNN confusion matrices, initial dataset (left), PCA dataset (right)

Chart

Description automatically generated

Figure .: MLP confusion matrices, initial dataset with standard scaler (left), initial dataset with minimax scaler (middle), PCA dataset (right)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Initial\_s & MLP** | **Initial\_m & MLP** | **PCA\_m & MLP** | **Initial\_m & DT** | **PCA\_m & DT** | **Initial\_m & SNN** | **PCA\_m & SNN** |
| **FAR** | 3.24 | 2.99 | 3.78 | 3.77 | 5.9 | 3.71 | 4.55 |
| **DR** | 75.38 | 74.69 | 75.55 | 71.74 | 72.8 | 73.72 | 71.79 |
| **PRECISION** | 96.85 | 97.06 | 96.35 | 96.18 | 94.22 | 96.33 | 95.42 |
| **RECALL** | 75.38 | 74.69 | 75.55 | 71.74 | 72.8 | 73.72 | 71.79 |
| **F1-SCORE** | 84.77 | 84.42 | 84.69 | 82.18 | 82.14 | 83.53 | 81.94 |
| **FPR** | 3.15 | 2.94 | 3.65 | 3.82 | 5.78 | 3.67 | 4.58 |
| **FNR** | 25.17 | 25.64 | 25.14 | 27.96 | 27.64 | 26.5 | 28.09 |

Table .: Performance evaluation of the best seven combinations

Table 4.8 shows the performance evaluation of the seven combination that performed better. The final conclusions for the binary classification experiment can be extracted. There is a relatively low false alarm rate (FAR) due to the factor that most of the times normal data is classified correctly. There false negative rate (FNR) is considerably high, but the reason is that sometimes malicious and normal data have almost same feature values. Moreover, it can be observed a high precision percentage meaning that if the model classified the data traffic as an attack there is a really high chance that it is correct (on average 96%). On the other hand there is a slight low recall percentage (detection rate) meaning that not all the malicious data can be classified correctly and thus detected. Overall, combining the precision and the recall (F1-score) the performance of the classifiers can be considered good enough.

## Multiclass Classification

As mentioned in Section 4.1 Table 4.2 the multiclass classification was made using 5 categories of data. It has to be mentioned that the initial types of categories in the training set were xx and in the test set were xx. Table xx shows in which major attack category each minor category belongs to.

|  |  |  |
| --- | --- | --- |
| Major Categories | Description | Minor Categories |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

The second experiment undertaken, was multiclass classification. Since it was already observed that the datasets with the best results were either the initial dataset having all the features or the reduced dataset using PCA, only those two datasets were used for this experiments. Furthermore only MLP, SNN and DT were used as well.

For every combination of model and dataset, 5 executions were made and the best results were reported.

Figure .: Multiclass classification accuracies of the seven combinations

|  |  |
| --- | --- |
| Combination | Accuracy |
| Initial\_m & MLP | 80.83 |
| PCA\_m & MLP | 79.65 |
| Initial\_s & MLP | 79.37 |
| Initial\_m & DT | 77.01 |
| PCA\_m & DT | 76.32 |
| Initial\_m & SNN | 76.15 |
| PCA\_m & SNN | 70.36 |

Table .: Multiclass classification accuracies of the seven combinations

Figure 4.5 and Table 4.9 show the accuracies of every combination used for creating a model in descending order. The best combination was when MLP were used with the full feature dataset normalized with the minmax scaler. On the other hand the worst performance was when SNNs were used with the PCA dataset normalized with the minmax scaler.

Chart

Description automatically generated

Figure .: Confusion matrix of Initial\_m and MLP combination

Chart

Description automatically generated

Figure .: Confusion matrix of Initial\_m and DT combination

Chart

Description automatically generated

Figure .: Confusion matrix of Initial\_m and SNN combination

Figures 4.6, 4.7 and 4.8 show the confusion matrices of the best performance for every ML technique used for this experiment. Again, we can extract a pattern by just observing the confusion matrices. Normal data is mostly classified correctly no matter the classifier used. Furthermore, DoS and Probe attacks can be classified correctly in a big percentage. On the other hand R2L and U2R types of attacks are too difficult to be correctly classified. Finally it can be said that R2L attack is almost always classified as normal activity which may mean that this kind of attack has really similar feature values as a normal instance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial M MLP | | | | |
| Classification Report | | | | |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **Normal** | 0.74 | 0.97 | 0.84 | 9711 |
| **DoS** | 0.96 | 0.84 | 0.9 | 7460 |
| **Probe** | 0.86 | 0.77 | 0.82 | 2421 |
| **R2L** | 0.56 | 0.22 | 0.31 | 2885 |
| **U2R** | 0.7 | 0.31 | 0.43 | 67 |
| **Average** | 0.77 | 0.62 | 0.66 | 22544 |

Table .: Initial\_m and MLP classification report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial M DT | | | | |
| Classification Report | | | | |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **Normal** | 0.7 | 0.97 | 0.81 | 9711 |
| **DoS** | 0.97 | 0.85 | 0.91 | 7460 |
| **Probe** | 0.84 | 0.79 | 0.81 | 2421 |
| **R2L** | 0.83 | 0.07 | 0.12 | 2885 |
| **U2R** | 0.88 | 0.22 | 0.36 | 67 |
| **Average** | 0.84 | 0.58 | 0.6 | 22544 |

Table .: Initial M and DT classification report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial M SNN | | | | |
| Classification Report | | | | |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **Normal** | 0.66 | 0.97 | 0.79 | 9711 |
| **DoS** | 0.96 | 0.82 | 0.88 | 7460 |
| **Probe** | 0.84 | 0.69 | 0.76 | 2421 |
| **R2L** | 0.32 | 0 | 0 | 2885 |
| **U2R** | 0 | 0 | 0 | 67 |
| **Average** | 0.56 | 0.5 | 0.49 | 22544 |

Table .: Initial M and SNN classification report

Tables 4.10, 4.11 and 4.12 shows the classification reports of the best performance for every ML technique. R2L and U2R types of attacks are really difficult to be classified correctly. The best average F1-score comes when using MLP and the least score when SNNs are utilised. Finally DoS attacks are the types of attacks easier to identify.

## NAN Federated Learning

The third part of experiments made were about utilising federated learning. More specifically three smaller experiments were undertaken. The first experiment was by using three different machine learning techniques (MLP, SNN,LR) and by varying the number of clients (SMs) observe the performance of the new centralized model. The second experiment uses the same three techniques but has only two clients. Those two clients have totally different data. The scope of the second experiment is to observe if a high performance model can be trained. Finally the third experiment was the same as the second experiment but this time the data were split over three clients.

There are multiple parameters that can be set in the server side involving the number of clients. Table 4.13 shows the variables set for the experiments made. The full list of parameters can be found at the Flower framework webpage [25]. Those parameters show the versatility of using federated learning, meaning that the number of clients needed for every round can vary (e.g. due to disconnection) and still be able to train the model. However the parameters were set fixed for the experiment because the behaviour of having the specific number of clients was desired to be observed.

|  |  |
| --- | --- |
| Variable | Description |
| fraction\_fit = 1 | Assigns the percentage (%) of available clients to be sampled in the next round, e.g. 1 = 100% |
| min\_fit\_clients=2,3,5,7 | Assigns the minimum number of clients to be sampled for the next round |
| min\_available\_clients=2,3,5,7 | Minimum number of clients that need to be connected to the server before a training round can start |
| min\_eval\_clients=2,3,5,7 | Minimum number of clients to be selected for evaluation |

Table .: Federated Learning parameters set

### Binary Classification

For this sub-experiment the number of clients used was two (2), three (3), five (5) or seven (7). For MLP and LR models the training took place for 20 rounds, whereas for SNN the training took place for only 5 rounds. The reason was because SNN tend to overfit really quick.

The results shown concern the performance of the IDS of the server side model created by averaging the weights of the clients models. The best results out of 3 executions are shown for each combination. Figure 4.9 shows a chart of accuracies for federated learning using multi-layer perceptron model. The black line indicates the average accuracy after each training round. It can be clearly seen that after just one round of training and calculating the average weights , the centralised model has a performance of almost 80% accuracy. Now comparing the different number of clients used , the best overall performance comes from 5 clients whereas the worst from 7 clients. Having 2 or 3 clients performs good enough as well.

Figure .: MLP Federated Learning Accuracy Chart

Figure .: LR Federated Learning Accuracy Chart

Figure 4.10 illustrates the accuracy of a federated learning model using logistic regression. It can be clearly observed that no matter the number of clients used the accuracies fluctuate at the same levels. Moreover there is an obvious peak after the second round of training and then a slight decline of performance, but after round 10 the performance is very steady. This can mean that using logistic regression for federated can be good if you want to apply it in a big scale.

Figure .: SNN Federated Learning Accuracy Chart

Figure 4.11 shows accuracy chart of when SNNs are used, again the best performance is observed after just one round of training. The accuracy of training for 3 and 5 clients reaches over 87% in the first round. As with MLP, having a big number of clients does not mean better performance, whereas when using LR the number of clients does not affect the performance much.

Figure .: Accuracy Comparison of ML techniques trained with 3 clients

Having 3 clients seemed to perform better than any other number of clients. Figure 4.12 shows a comparison between the three techniques. It can be observed that the highest accuracy comes from SNN in just the first training round, but then the SNN accuracy starts decreasing. Furthermore, MLP have the overall best performance and LR have the overall worst performance. The performance of the SNN after the first training round is shown in Table 4.14 and the confusion matrix is shown in Figure 4.13.

|  |  |
| --- | --- |
| Metric | Percent |
| Accuracy | 87.39347 |
| Precision | 90.71854 |
| Recall | 86.72904 |
| F1\_score | 88.67894 |
| Detection\_rate | 86.72904 |
| FAR | 11.72833 |
| FPR | 9.281462 |
| FNR | 16.57738 |

Table .: Performance Evaluation of SNN trained with 3 clients after the 1st round of training

Chart, treemap chart

Description automatically generated

Figure .: Confusion Matrix of SNN trained with 3 clients after the 1st round of training

Reaching an accuracy of 87.4% and an F1-score of 88.7% it can be concluded that creating an IDS using multiple clients/SMs can be more beneficial than when having just a standalone model. Furthermore it shows that FL can still achieve and trained a high performance model even though the data are stored in multiple locations.

### 2 Clients Experiment

For this experiment, the training dataset was split into two subsets. The two subsets had no common attack type records. Client 1 had normal data and malicious data belonging into the DoS category, Client 2 had normal data and malicious data belonging to Probe, R2L and U2R attack types. Table 4.15 shows in more detail how the train data were assign.

|  |  |  |
| --- | --- | --- |
|  | **Client 1** | **Client 2** |
| **Normal Records** | 33671 | 33672 |
| **Attack Records** | 45927 | 12703 |
| **Total** | **79598** | **46375** |

Table .: Dataset break-down for each of the 2 Clients

Three executions were made for each machine learning technique and the best results were reported. Starting with comparing the accuracies of the three machine learning techniques shown in Figure 4.14.

Figure .: Accuracy comparison of ML Techniques when trained with the 2 Clients dataset

As shown in Figure 4.14 the highest accuracy achieved by any of the three ML models is 78.4% (using MLP). From this result it can be extracted that when the datasets of the clients used for training are completely different, then the centralised model performance is a bit lower than when there are common attacks in the data. This may happen because the FedAVG method is used which just computes the average weights of all the clients.

Table 4.16 shows a comparison of the highest accuracies achieved when the 2 Clients have common data and when they don’t.

|  |  |  |
| --- | --- | --- |
|  | Common Data | Different Data |
| **MLP** | 82.57% | 78.44% |
| **SNN** | 77.18% | 74.18% |
| **LR** | 75.77% | 73.16% |

Table .: Comparison

Furthermore it can be observed for SNNs that as the training rounds go by the performance starts to become poorer. On the other hand LR the performance fluctuates a bit but stays between a certain level. Finally MLP performance increases as the training rounds increase as well.

Each client had as a cross validation set the training set of the other client, in order to see how well it can predict the attack types that are not in their training set. Figures 4.15, 4.16 and 4.17 show the cross validation results of the MLP model, the SNN model and the LR model respectively.

Figure .: MLP Cross Validation accuracy for each of the 2 Clients

It can be clearly extracted that for all the ML techniques Client’s 2 cross validation accuracy is very high from even the first training round. From this it can be asserted that Client’s 1 data can be predicted easier than Client’s 2 data. Furthermore, Client’s 1 cross validation accuracy increases as the training rounds increase as well, for the MLP and SNN models, meaning that the weight updates from the server-side are useful. In addition, when observing the results from the Multiclass classification experiment DoS attacks are the easiest to be classified correctly, thus this is another reason that Client’s 1 data have such high cross validation accuracy.

Figure .: SNN Cross Validation accuracy for each of the 2 Clients

Figure .: LR Cross Validation accuracy for each of the 2 Clients

### 3 Clients Experiment

For this experiment, the training dataset was split into three sub-datasets. Each dataset contains no common attack types as the other two. Table 4.16 shows how the training data attack types were split to each client. Table 4.17 shows how many records each client has.

|  |  |  |
| --- | --- | --- |
| Client 1 | Client 2 | Client 3 |
| Neptune | Satan | Ipsweep |
|  | Smurf | Protsweep |
|  | Nmap | Back |
|  | Warezclient | Teardrop |
|  | Pod | Guess-passwd |
|  | Land | Buffer\_overflow |
|  | Imap | Warezmaster |
|  | Loadmodule | Rootkit |
|  | Multihop | Ftp-write |
|  | Perl | Phf |
|  |  | Spy |

Table .: Illustration of how training data normal and attack types was assigned to each client

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Client 1* | *Client 2* | *Client 3* |
| *Normal Records* | 22448 | 22447 | 22448 |
| *Attack Records* | 41214 | 8911 | 8505 |
| *Total* | **63662** | **31358** | **30953** |

Table .: Dataset break-down for each of the 3 Clients

Figure .: Accuracy comparison of ML Techniques when trained with the 3 Clients dataset

Figure .: MLP Cross Validation accuracy for each of the 3 Clients

Figure .: SNN Cross Validation accuracy for each of the 3 Clients

Figure .: LR Cross Validation accuracy for each of the 3 Clients

## Summary

# Conclusion and Future work

# References

|  |  |
| --- | --- |
| [1] | M. A. Faisal, Z. Aung, J. R. Williams and A. Sanchez, “Data-Stream-Based Intrusion Detection System for Advance Metering Infrastracture in Smart Grid: A Feasibility Study,” *IEEE Systems Journal,* vol. 9, no. 1, pp. 31-44, March 2015. |
| [2] | C. Lo and N. Ansari, “CONSUMER: A Novel Hybrid Intrusion Detection System for Distribution Networks in Smart Grid,” *IEEE Transactions on Emerging Topics in Computing,* vol. 1, no. 1, pp. 33-44, June 2013. |
| [3] | M. Brain and D. Ross, “How Power Grids Work,” [Online]. Available: https://science.howstuffworks.com/environmental/energy/power6.htm. [Accessed March 2022]. |
| [4] | X. Fang, S. Misra, G. Xue and D. Yang, “Smart Grid-- The New and Improved Power Grid: A Survey,” *IEEE Communications Surveys & Tutorials,* vol. 14, no. 4, pp. 944-980, 2012. |
| [5] | V. C. Gunor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati and G. P. Hancke, “Smart Grid Technologies: Communication Technologies and Standards,” *IEEE Transactions on Industrial Informatics,* vol. 7, no. 4, pp. 529-539, 2011. |
| [6] | P. Kumar, Y. Lin, G. Bai, A. Paverd, J. S. Dong and A. Martin, “Smart Gird Metering Networks: A Survey on Security, Privacy and Open Research Issues,” *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, pp. 2886-2927, 2019. |
| [7] | Wikipedia, “Smart Grids,” [Online]. Available: https://en.wikipedia.org/wiki/Smart\_grid. [Accessed March 2022]. |
| [8] | I. S. Jha, S. Sen and V. Agarwal, “Advance Metering Infrastructure Analytics - A Case Study,” *Eighteenth National Power Systems Conference (NPSC),* pp. 1-6, 2014. |
| [9] | H.-J. Liao, C.-H. R. Lin, Y.-C. Lin and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *Journal of Network and Computer Applications,* vol. 36, no. 1, pp. 16-24, 2013. |
| [10] | M. Ozkan-Okay, R. Samet, O. Aslan and D. Gupta, “A Comprehensive Systematic Literature Review on Intrusion Detection Systems,” *IEEE Access,* vol. 9, pp. 157727-157760, 20221. |
| [11] | A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity,* vol. 2, no. 1, 2019. |
| [12] | Q. S. Qassim, A. M. Zin and M. J. A. Aziz, “Anomalies Classification Approach for Network-based Intrusion Detection System,” *International Journal of Network Security,* vol. 18, no. 6, pp. 1159-1172, 2016. |
| [13] | P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez and E. Vazques, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers&Security,* vol. 28, no. 1-2, pp. 18-28, 2009. |
| [14] | K. Scarfone and P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS), National Institude of Standards and Technology (NIST), 2007. |
| [15] | I. Ullah and Q. H. Mahmoud, “An intrusion detection framework for the smart grid,” *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE),* pp. 1-5, 2017. |
| [16] | S. Khan, K. Kifayat, A. K. Bashir, A. Gurtov and M. Hassan, “Intelligent intrusion detection system in smart grid using computational intellignece and machine learning,” *Transaction on Emerging Telecommunications Technologies,* vol. 32, no. 6, 2020. |
| [17] | J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong and W. D. Lu, “Training Spiking Neural Networks Using Lessons From Deep Learning,” *arXiv preprint arXiv:2109.12894,* 2021. |
| [18] | T. Yiu, "Understanding Random Forest," Toward Data Science, 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed 22 July 2022]. |
| [19] | V. Aliyev, “Gradient Boosting Classification explained through python,” September 2020. [Online]. Available: https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d. [Accessed 22 July 2022]. |
| [20] | A. Pal, “Gradient Boosting Trees for Classification: A Beginner's Guide,” October 2020. [Online]. Available: https://medium.com/swlh/gradient-boosting-trees-for-classification-a-beginners-guide-596b594a14ea. [Accessed 22 July 2022]. |
| [21] | Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu and B. He, “A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection,” *IEEE Transactions on Knowledge and Data Engineering,* 2021. |
| [22] | J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh and D. Bacon, “Federated Learning: Strategies for Improving Communication Efficiency,” *NIPS Workshop on Private Multi-Party Machine Learning,* 2016. |
| [23] | Q. Yang, Y. Liu, T. Chen and Y. Tong, “Federated Machine Learning: Concept and Applications,” *ACM Transactions on Intelligent Systems and Technology,* vol. 10, no. 2, pp. 1-19, 2019. |
| [24] | F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher and M. Perrot, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research,* vol. 12, pp. 2825--2830, 2011. |
| [25] | B. Daniel, T. Taner, M. Akhil, Q. Xinchi, P. Titouan and L. Nicholas, “Flower: A Friendly Federated Learning Research Framework,” 2020. |
| [26] | M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications,* pp. 1-6, 2009. |
| [27] | T. Su, H. Sun, J. Zhu, S. Wang and Y. Li, “BAT: Deep Learning Methods onNetwork Intrusion Detection usingNSL-KDD dataset,” *IEEE Access,* 2020. |
| [28] | R. Pramoditha, “11 Dimensionality reduction techniques you should know in 2021,” Towards Data Science, 14 April 2021. [Online]. Available: https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b. [Accessed April 2022]. |

Appendix 1