INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Deep Learning Model Transposition for Network Intrusion Detection Systems

João Pedro da Mota Pereira de Figueiredo

Master in Computer Engineering

Supervisors:
PhD Carlos José Corredoura Serrão, Associate Professor, ISCTE-IUL

PhD Ana Maria Carvalho de Almeida, Associate Professor, ISCTE-IUL

November, 2022

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

Deep Learning Model Transposition for Network Intrusion Detection Systems

João Pedro da Mota Pereira de Figueiredo

Master in Computer Engineering

Supervisors:
PhD Carlos José Corredoura Serrão, Associate Professor, ISCTE-IUL

PhD Ana Maria Carvalho de Almeida, Associate Professor, ISCTE-IUL

November, 2022

Direitos de cópia ou Copyright
©Copyright: João Pedro da Mota Pereira de Figueiredo.

O Iscte - Instituto Universitário de Lisboa tem o direito, perpétuo e sem limites geográficos, de arquivar e publicitar este trabalho através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, de o divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Acknowledgements

First and foremost, I want to acknowledge the everlasting love and support from my family, as they are the reason behind everything that I do.

Secondly, I want to thank my supervisors for their wisdom and support in all of this process. As a working student, I don't always have the best schedules. Their endless understanding and support were vital for the success of this work.

Resumo

As empresas procuram promover uma rápida digitalização de seus processos de negócio e novas funcionalidades disruptivas para obter vantagens competitivas sobre os seus concorrentes. Geralmente, isto resulta numa superfície de ataque mais ampla que atrai a exploração de atacantes. Como os orçamentos são escassos, uma das soluções de segurança mais populares que as empresas escolhem para investir é em NIDS. Como os NIDSs baseados em anomalias trabalham sobre uma baseline de atividade normal, uma das principais áreas de desenvolvimento é o treino de modelos de deep learning robustos o suficiente para que, dado um contexto de rede diferente, o sistema seja capaz de identificar com uma alta taxa precisão alguma intrusão. Neste estudo, propomos um NIDS baseado em anomalias usando um modelo empilhado de LSTMs de deep learning com uma nova técnica de pré-processamento que fornece features livres de contexto e supera a maioria dos trabalhos relacionados, obtendo mais de 99% de precisão sobre o dataset CICIDS2017. Também pode ser aplicado em diferentes ambientes de rede sem perder a precisão, pois utiliza features livres de contexto. Além disso, usando ataques de rede simulados, o nosso NIDS consegue detectar categorias específicas de intrusões.

Palavras-Chave: sistemas de detecção de intrusão de rede; detecção de intrusão; detecção de anomalias; aprendizagem profunda; memória de curto prazo longa.

Abstract

Companies seek to promote a swift digitalization of their business processes and new disruptive features to gain advantage over their competitors. This often results in wider attack surface that attract attack exploitation. As budgets are thin, one of the most popular security solutions CISOs choose to invest is in NIDS. As anomaly-based NIDSs work over a baseline of normal and expected activity, one of the key areas of development is the training of deep learning classification models robust enough so that, given a different network context, the system is still capable of high-rate accuracy for intrusion detection. In this study, we propose an anomaly-based NIDS using a deep learning stacked-LSTM model with a novel pre-processing technique that gives it context-free features and outperforms most related works, obtaining over 99% accuracy over the CICIDS2017 dataset. It can also be applied to different environments without losing its accuracy since it uses context-free features. Moreover, using synthetic network attacks, our NIDS can detect specific categories of attacks.

Keywords: network intrusion detection system (NIDS); intrusion detection; anomaly detection; deep learning (DL); long short-term memory (LSTM).

General Index

Acknowledgements ..... **i**
Abstract ..... **ii**
General Index ..... **iv**
Table Index ..... **v**
Figure Index ..... **vi**
Glossary ..... **vii**
Chapter 1 – Introduction **1**
Chapter 2 – Background Study **4**
2.1. Intrusion Detection System 4
2.2. Machine Learning, Deep Learning and Intrusion Detection 5
2.3. Advanced Anomaly-Based Intrusion Detection using Long Short-Term Memory Networks 8
Chapter 3 – Methodology **15**
3.1. Building the NIDS 15
3.1.1. Dataset 16
3.1.2. Pre-processing 17
3.1.3. Stacked LSTM Model 21
3.2. Running real-world data through the NIDS 22
3.2.1. Real-world network traffic data recording 23
3.2.2. Feeding the captured data to the NIDS 25
Chapter 4 – Results and Discussion **26**
4.1. CICIDS2017 evaluation 26
4.2. Real-world network traffic data evaluation 28
Chapter 5 – Conclusions and Future Works **30**
References **32**
Annex A – CICIDS2017 extracted features **39**

Table Index

Table 1 – CICIDS2017 dataset contents 16
Table 2 – Recorded network use cases 24
Table 3 – Evaluation metrics 27
Table 4 – Comparison with similar studies 27
Table 5 – Real-world network data captured 28

Figure Index

Figure 1 – Anomaly-based DL NIDS architecture 15
Figure 2 – One-Hot Encoding explanation 20
Figure 3 – Stacked LSTM model topology 21
Figure 4 – Confusion matrix with our results 27

Glossary

LSTM – Long Short-Term Memory

IDS – Intrusion Detection System

NIDS – Network Intrusion Detection System

DL – Deep Learning

ML – Machine Learning

ANN – Artificial Neural Network

RNN – Recurrent Neural Network

CNN – Convoluted Neural Network

GRU – Gated Recurrent Unit

DoS – Denial of Service

DDoS – Distributed Denial of Service

KNN – K Nearest Neighbour

SVM – Support Vector Machine

MLP – Multi-Layer Perceptron

IOC – Indicator of Compromise

CISO – Chief Information Security Officer

IoT – Internet of Things

TP – True Positive

TN – True Negative

FP – False Positive

FN – False Negative

Chapter 1 – Introduction

As companies seek to find competitive advantages over their competitors with the digitalization of their business processes and new disruptive features are made possible by implementing emerging technologies, more and more attackers are seeking to find gaps in companies' products for financial gain [1]. Unfortunately, companies often overlook security in the design of their products and services in the name of faster releases to production of their new features – this often results in a bigger attack surface for attackers to explore. Consequently, and thanks to new regulatory and compliance requirements, legal obligations and governmental pressure, more and more companies are starting to invest in cybersecurity [2]. This usually includes assigning the organizational cybersecurity function of the company to a specific role – usually the Chief Information Security Officer (CISO).

In today's scenario, where companies must carefully choose where to spend their budget to prevent an ever-growing number of attackers looking to exploit their company's products and services – every choice matters. One of the most popular security solutions companies choose to invest in, is in Intrusion Detection and Prevention Systems. An Intrusion Detection System (IDS) [3] is a security product that actively monitors either hosts (as a Host Intrusion Detection System, HIDS) or networks (as a Network Intrusion Detection System, NIDS) and detects malicious activity. The difference between the Intrusion Detection Systems and Intrusion Prevention Systems, is that the latter go a step beyond and try to block the detected intrusions. While both systems have their specific uses, we focus on the IDS as we are specifically interested in the detection capabilities of these systems. Typically, they have 2 detections methods: signature-based and anomaly-based [4].

The Signature-based detection method is the most common detection method used by IDSs. This detection method consists in searching for Indicators of Compromise (IOC) in either the host or the network. These can range from a series of commands being ran in sequence, a known malware file or file hash, or even a sequence of bits in a network packet. More commonly, signature-based detection methods are used by most anti-virus solutions and they're the reason why anti-virus need to be updated consistently – they're as efficient as the signatures they can recognize. Just as most anti-virus, signature-based IDSs need to be consistently updated in order to protect companies against intrusions performed via new attacks.

On the other hand, the anomaly-based detection method is based on heuristics and rules. Instead of looking for specific signatures or patterns, they start by measuring and defining a baseline of expected activity in either the host or network and, after establishing the baseline, they look for outliers. Since anomaly-based IDSs are trained to recognize normal activity and behavior of their respective hosts or networks, they don't need to be constantly updated to fulfill their purpose. This makes them especially relevant in detecting intrusions performed by exploiting 0-day vulnerabilities or in legacy, outdated or non-updated systems that lack the prevention capabilities to defend against known vulnerabilities.

Although anomaly-based IDS might have an advantage in detecting sophisticated new attacks, they're plagued by a high false-positive detection rate and the ability to be fooled by a correctly delivered attack [5]. For this reason, most security companies with actively developed IDSs implement both detection methods in their products – signature-based and anomaly-based.

Coincidently, as more companies further harden their security updates procedure and technological planned obsolescence policies [6], there has been a shift in the cyber-attack surface exposure – while previously exposure came from unpatched vulnerabilities, today is coming from 0-day vulnerabilities [7]. As such, being the most appropriate detection method for identifying attacks performed by exploiting a 0-day vulnerability, there has never been more interest than now in further developing anomaly-based detection IDSs.

In accordance with this need of further development there was a major shift in the way anomaly-based detection mechanisms work. In the early days, most would work with a strict mathematical model and would flag any deviation as an attack [4]. With the recent availability of new and better machine learning models, such as artificial neural networks, more and more of anomaly-based IDS solutions started to adopt machine learning for their classification mechanisms. The usage of several artificial neural network classes for intrusion classification, such as recurrent neural networks (RNN), and convolutional neural networks (CNN), has been successful [8] [9] [10].

Since anomaly-based detection mechanisms work over a baseline of normal and expected activity, they are susceptible to context problems, as machine learning classification models seldom maintain accuracy when used in different networks. In this study, we investigate different techniques and approaches to address this problem through the development of an anomaly-based DL NIDS, with the objective that given a robust

pre-processed dataset and context-free features, the DL NIDS can be used in different environments without losing its accuracy of intrusion detection.

We aim to answer the following research questions:

- How accurate is a DL model in intrusion detection?
- How can we transpose a simulated dataset-trained DL NIDS to a real word network environment?

The key contributions of this thesis are as follows:

- A novel pre-processing technique that improves the detection performance of an anomaly-based DL NIDS in a real-world network environment.
- A highly accurate anomaly-based DL NIDS consisting of a stacked LSTM architecture.
- An analysis of the impact in implementing an anomaly-based DL NIDS in a real-world network environment.

We start our work by reviewing the relevant literature, then we describe our proposed methodology to tackle the described problem and accomplish our objectives, afterwards we measure, describe and discuss the results we obtained using our methodology, and we end the work with a conclusion where we identify relevant information for future works.

Chapter 2 – Background Study

In this section we perform a systematic review of the relevant literature in order to observe the current state of the art on our research topic.

We identified the relevant literature via advanced search on IEEE Xplore and Google Scholar for the terms: “intrusion detection”, “network intrusion detection”, “anomaly detection”, “deep learning” and “long short-term memory”.

2.1. Intrusion Detection System

Since the inception of the first IDS in 1980 by James P. Anderson [3] for the National Security Agency (NSA), a great deal of research has been done on its evolution to match todays reality of network infrastructures. This first model strictly used signature-based detection and was designed to help system administrators review audit trails.

In 1986, Dorothy E. Denning et al. [4], proposed an evolution of James P. Anderson’s IDS model using both signature-based detection and anomaly-based detection methods. The authors proposed several statistical approaches for anomaly detection, but most implementations used a Markov Chain model.

A few years later, in 1992 , and with the increasing prominence of malware usage on information systems, Wang et al. [8] proposed an anomaly-based detection method for IDS using a one-class Support Vector Machine (SVM) in conjunction with a Markov Chain model, which stands as one of the first examples of using machine learning (ML) in intrusion detection. The results were not only a more accurate detection method for anomaly detection, but also a more efficient model to train.

In the following years most works focused on advancing the anomaly-based detection capabilities, but most of the models still used statistical approaches. In 2009, Chih-Fong Tsai et al. [11] proposed an IDS model using a hybrid learning model called triangle area nearest neighbors (TANN), which is composed of three stages: extraction of cluster centers with a k-means algorithm, new data formulation by triangle areas, and application of a KNN classifier on the data. Using this model, the authors achieved a 97.0% accuracy rate.

Although in recent years most researchers have shifted to other kinds of algorithms for intrusion detection, which we will detail next, some authors continued to investigate and compare different regression analysis algorithms.

Tahir Mehmood et al. [12] compared SVM, Naïve Bayes, J48 (a type of Decision tree) and Decision table algorithms. The authors then compared each model against four different kinds of attacks: denial of service, remote to local, user to root and probe attack. While none of the models showed a high true positive rate for every kind of attack, the J.48 performed better than the adversaries while presenting a low misclassification rate.

Zecheng He et al. [13] focused on using ML for DDoS coming from the cloud detection in its research. With this purpose, the authors compared different ML models against four kinds of DDoS attacks: SSH brute-force attack, DNS reflection attack, ICMP flood and TCP SYN attack. From the nine models used by the authors, the one that performed better was the SVM.

2.2. Machine Learning, Deep Learning and Intrusion Detection

Deep learning (DL) is a sub-field of ML that models the learning process using multiple layers of neurons forming an artificial neural network (ANN). The word "deep" is referring to the huge number of layers through which the data is transformed, and it was coined by Rina Dechter [14] in 1986. While DL was fairly researched, it was only by 2012 that it started to be implemented to solve a variety of tasks. This happened mostly due to three reasons:

1. Advances to the algorithms, which enabled them to be used on a broader range of domains.
2. Increased availability of bigger datasets, which were used to train the models.
3. Increased computing power, which enabled more organizations and individuals to use DL systems. One key development for this was made by a NVIDIA release of a new graphical processing unit (GPU) in 2009 that could be used to increase the speed of DL systems by 100 times [15].

It's also important to note that several kinds of ANNs are used for DL. Some of the most known are multilayer perceptron (MLP), convolutional neural networks (CNN) and recurrent neural networks (RNN).

As the scene was set for further research in how to apply DL to more domains, it was not long after that we started to see a new variant of anomaly-based detection methods for IDS.One of the first integrations of an ANN for anomaly-based NIDS was published

in 2008 by Jimmy Shum et al. [9]. The data for the proposed model was obtained from the DARPA repository and consisted of nine weeks of network traffic data recorded at the MIT Lincoln Laboratory. This dataset is commonly known as the KDD CUP 99 dataset. Instead of measuring the accuracy of the model by the network packet, the authors measured it by verifying if the model could detect that an attack was happening, even if it wassn't able to pinpoint if the exact packets are benign or an attack. While using this method can provide a greater level of accuracy in identifying if an attack is happening or not, the only plausible reaction to a detected attack wass that of notifying a system administrator so he can manually investigate the matter. Nevertheless, the experiment was successful at the time and the model achieved 100% accuracy for attack detection of known attack sets according to the author's definition of an accurate classification.

In 2011, Xianjin Fang et al. [10] integrated an ANN in the popular open-source IDS "Snort". In order to achieve this integration, Xianjin Fang et al. developed a custom detection plug-in for the IDS detection engine that, depending on the ANN model classification, could generate an alert file for the system's administrator. Despite being able to develop the custom plug-in, the authors didn't train or test the model and as such there isn't a recorded accuracy for this implementation.

In 2015, Jamal Esmaily et al. [16] proposed an IDS model but with a hybrid approach using both MLP and DT for classification, and achieved 99.7% accuracy on the KDD CUP 99 dataset. Because most of the data was captured by listening to traffic in a short period of time, the majority of the records in the dataset are exactly identical. To train the model more efficiently, the authors deleted the redundant records. The model consisted of two phases: in the first phase, the dataset was trained on both an MLP and a DT model and outputted a binary classification of either "attack" or "normal", after this, the resulting output of both models was processed by another MLP network that outputted the final result of again, either "attack" or "normal".

In 2018, Nikita Lyamin et al. [17] documented the challenges of using DL for intrusion detection in a VANET, which is a mobile ad hoc network applied to the context of vehicles. Despite acknowledging that existing protocols in vehicle-to-vehicle communications integrated in a DL model are capable of detecting a DoS attack, they conclude that it will be difficult to train such a model because there isn't enough available data to train it and, even if there was, most DL methods require annotated datasets pertaining to a DoS class.

In a comprehensive study of different ML approaches to intrusion detection, Kunal et al. [18] surveyed 19 different models by different authors in order to evaluate which one has had the best outcome. While not being able to identity a correlation between the different aspects of each model and their outcome, the author did document several different possible approaches for further works.

Internet of Things (IoT) has been increasingly adopted in modern households, as modern appliances are being designed to improve day-to-day activities by communicating with each other. The resulting network of these appliances can be called a Web of Things, and consists of appliances communicating via web standards, such as REST APIs. Emmanuel Alalade et al. [19] proposed in 2020 an IDS for Smart Homes using artificial immune system and extreme learning hybrid approach (AIS-ELM). The AIS was used for input optimization and pre-processing, and ELM for classification. In the authors framework, the resulting action was a notification to an arbitrary email in order to locally react to an attack.

As denials of service (DoS) and distributed denial of service (DDoS) attacks are among the most regularly performed and heavily investigated attacks, a lot of authors have been releasing new methodologies able to detect this attacks.

Focusing on a specific kind of a network attack, Irfan Sofi et al. [20] explored the capabilities of four different ML algorithms (Naïve Bayes, DT, MLP and SVM) for the detection and analysis of modern types of DDoS attacks in 2017. The authors found that there wasn't any public dataset available for training and testing a model again modern DDoS attacks. Thus, they recorded their own dataset in a controlled environment. While this enabled the training and testing of a model, the volume of records was low – in total, the dataset has approximately 75 000 records, where approximately 67 000 are "normal" records. In comparison, the KDD CUP 99 dataset, which is still widely used by research for IDS models, has 5 000 000 records. The results of the comparison were that the MLP achieved the highest accuracy rate of the four models, with a 98.9% accuracy rate.

Also focused on DDoS attack detection, Merlin Dennis et al. [21] proposed two approaches for detection of these attacks in a software defined network (SDN). The first one was a statistical approach that implemented entropy computation and flow statistics analysis, and the second one was a ML approach that used Random Forest classifier for detection. In the end, the first approach obtained a 92% accuracy, and the second achieved 97.7% accuracy. The authors also compared their ML approach against similar ones and

accredited their success to the random forest classifier, which has fast computing times and resistance to noisy data, and the implementation of weighted voting instead of the standard majority voting. Lingfeng Yang et al. [22] also proposed an SDN framework capable of identifying and defending the network against DDoS attacks. For this, the authors defined a framework consisting of 3 parts: the traffic collection module, the DDoS attack identification module, and the flow table delivery module. The DDoS attack identification module resorted to a SVM network to identity the attacks and model was trained with the KDD CUP 99 dataset, obtaining a 99% accuracy. Abdulsalam Alzahrani et al. [23] went on a different direction than the previous authors that worked on SDN, and proposed an IDS as part of the SDN controller to detect pure intrusions on the network instead of detecting DDoS attacks. After evaluating and comparing different ML algorithms, the authors concluded that XGBoost, which is a tree-based method, was the more accurate one with 95.55%.

In a different approach, Pablo Rivas et al. [24] leveraged data from a honeypot network to train two models, an SVM and a CNN, in order to be able to detect DDoS attacks. The work published by the authors followed the Marist College's effort to develop, deploy and maintain a HoneyNet – a network of honeypots – called Peitho (named after a Greek god). This network is then able to attract malicious actors that promptly try to attack it, sometimes even using 0-day vulnerabilities. All of the network data captured by Peitho is then fed to an SVM model a CNN model in order to compare the different accuracy results. The CNN performed better, achieving results of 100% accuracy in identifying DDoS attacks. Another example of an effective ML model for DDoS detection was proposed by Jian Zhang et al. [25] with an approach of a gradient decision boosting tree (GDBT) in order to continuously evaluate and optimize feature selection in a closed loop.

2.3. Advanced Anomaly-Based Intrusion Detection using Long Short-Term Memory Networks

As further researchers compared several different kinds of DL algorithms for intrusion detection, one of them started to stand out – the long short-term memory network (LSTM).

Proposed initially by Hochreiter et al. in 1995 as a subset of the RNN [26], the name LSTM refers to the analogy that while a standard RNN has both long-term and short-term

memory, the LSTM architecture provides a short-term memory for RNNs that can last several timesteps, thus the name long short-term memory.

The authors published the LSTM model as a proposal to solve the vanishing gradient problem, which happens when training an ANN with backpropagation. Neural network's weights receive an update proportionate to the partial derivative of the error function with respect to the current weight during each training iteration. The issue is that the gradient may be vanishingly small in some situations, thereby preventing the weight from changing its value. In the worst-case scenario, this could prevent the neural network from further training. LSTM fight this problem by allowing gradients to flow unchanged while backpropagating. The result is a model that is capable of handling long-term dependencies.

LSTMs were a big step forward for what we could accomplish with RNNs. Xu et al. [27] used an LSTM model in order to create descriptions for images. In 2015, Google [28] started using LSTM models for its speech recognition services. More famously, OpenAI and DeepMind, two of the most prominent organizations for artificial intelligence research, started implementing LSTM models in order to beat humans at the video games Dota 2 and Starcraft II, respectively.

Following the ever-bigger trend of using LSTMs in DL problems, researchers started to document its effects for intrusion detection. With the objective of researching the trends ML and DL algorithms applied to intrusion detection, M. Stampar et al. [29] gathered and measured the number of publications pertained to intrusion and detection and artificial intelligence/machine learning in the period 2010-2014. The most common models they found were: NN, Genetic algorithm, Fuzzy logic, KNN, Naïve Bayes, Decision Tree, Random Forest, SVM and Self-organizing map. The results revealed that the most researched model was by far the NN with close to 1500 publications per year, having almost the double of the publications of the second most researched, the SVM with close to 800 publications per year.

In 2012, Pankaj Malhotra et al. [30] developed a model using stacked LSTM networks in order to predict anomalies in four different problems: pre-ventricular contractions in an electrocardiogram dataset, anomalies in power delivery of the space shuttle Marotta valve dataset, peaks in an electrical power demand dataset and faults of a multi-sensor engine dataset. While none of these anomalies are related to intrusion detection per se, an intrusion detection is in fact an anomaly in the context of the regular activity of a system

or a network. The authors concluded that a model consisting of stacking LSTM networks for anomaly detection is able to learn higher-level temporal patterns without previous knowledge of the pattern duration and that this model can be used for anomaly detection.

In 2018, Ali Mirza et al. [31] researched different kinds of LSTM networks for intrusion detection, such as vanilla LSTM, Bi-LSTM (which is a bidirectional LSTM network) and Deep Auto LSTM (which the authors define as a network with 5 stacks of LSTMs). The authors then trained and tested the models against the ISCX IDS 2012 dataset and concluded that the best performing model was the Deep Auto LSTM, with a F1-Score of 85%. In a similar fashion, Omar Alsyaibani et al. [32] also proposed a bidirectional LSTM network for intrusion detection. The authors then iterated over 24 cycles different training parameters in order to find the most suitable configuration and trained the model for 100 epochs on the CICIDS2017 dataset, reaching an F1-Score of 97.75%. After 1000 epochs, the model reached a 98.37% F1-Score. Monika Rokade et al. [33] proposed a similar framework to the previous authors, achieving a 95% accuracy with a NB ANN, RF and SVM hybrid model. Sara Althubiti et al. [34] experimented with a different dataset, the CIDDS-001, against a single LSTM layer model for intrusion detection and achieved a lower than usual accuracy of 84.83%.

Jorge Meira [35] compared four different intrusion detection techniques against the two most used dataset, the KDD CUP 99 and the ISCX 2012 IDS. The techniques used were isolation forest, equal frequency with z-score and autoencoder, z-score with nearest neighbor and z-score with K-means. While on the KDD CUP 99 dataset none of the techniques were particularly better than the others, in the ISCX 2012 IDS dataset the z-score with nearest neighbor performed significantly better than the competition.

In 2019, Thi-Thu-Huong Le et al. [36] proposed a novel IDS framework in order to reduce the high false-positives rates that plagued the anomaly-based IDSs. The framework proposed by the authors included a hybrid sequence forward selection algorithm with decision tree to generate the best feature set subset from the original feature set, then ran those features against several models of RNNs such as a traditional RNN, an LSTM and a GRU and finally evaluate the resulting performance from the different models. The models were trained with both the NSL-KDD 2010 IDS dataset and the ISCX 2012 IDS dataset. In the end, the RNN model with the best performance was the LSTM one, achieving accuracy of 92% with the NSL-KDD dataset and 97.5% with the ISCX dataset.

In 2020, Shinly Swarna Sugi et al. [37] researched different DL algorithms for intrusion detection in IoT networks. The authors built two models, one using LSTM and another using KNN, and compared the performance of both of them on detection time, kappa statistic, geometric mean, and sensitivity. Using the Bot IoT dataset, the LSTM model outperformed the KNN model in all counts. In the same year, Shin Park et al. [38] also proposed an LSTM model for intrusion detection in an IoT context network, but for classification the authors compared the predicted network packet against the actual packet using cosine similarity and Euclidean distance as scoring functions, achieving greater results than other intrusion detection techniques in similar contexts. In 2021, Jinsi Jose et al. [39] also researched the performance of different ML and DL algorithms for intrusion detection in IoT environments and concluded that DL algorithms for intrusion detection should be the focus of future works in this domain. In 2022, Ghada Abdelmoumin et al. [40] examined the usage of optimization techniques to enhance the performance of anomaly-based IDSs in an IoT context. For this, the authors tuned the hyperparameters via a procedural iteration of multiple combination of these parameters. For each combination, the authors evaluate the accuracy and other metrics in order to determine the optimal model. Still in 2022, Kumar Saurabh et al. [41] proposed a DL model consisting of stacked and bidirectional LSTMs for intrusion detection in an IoT environment. Running the model against the Bot IoT dataset, the authors achieved 99.99% accuracy. Sylvia Azumah et al. [42] also proposed an LSTM-based IDS model for IoT networks, but using a single LSTM layer network, achieving a 98% accuracy which is lower than the previous author's model that used stacked bidirectional LSTMs.

One of the biggest challenges of a NIDS, is processing the sheer dimension of a typical high network traffic dataset. Liang Zhang et al. [43] proposed in 2020 an improved LSTM IDS model based on a two-phase DL algorithm, where the first phase uses quantum particle swarm optimization (QPSO) in order to do feature selection and reduce the dimensionality, and the second phase uses an LSTM network for classification and therefore, intrusion detection. Using this model, researchers can better avoid dimensional disasters. Running it against the KDD CUP 99 dataset, the model's QPSO reduced the 41 features to 24 features, resulting in an upgrade of the LSTM from 83.47% to 97.79%, and an F1-Score of 96.65%. The authors affirm that the proposed model can be used in real network environments for intrusion detection.

Arunavo Dey [44] proposed in 2020 a CNN-LSTM hybrid model for intrusion detection and trained it on the CICS ISCX IDS 2018 dataset. The author's model was able to outperform similar works done before and achieved 99.98% accuracy. In order to achieve these results, model had four different layers: a CNN layer, that performed the convolution operation on the input, an LSTM layer, an attention layer and finally a dense layer. S Amutha et al. [45] proposed a similar IDS model that resourced purely to an LSTM network for classification and used the UNSW NB 18 IDS dataset for training and testing. The authors achieved a similar accuracy of the previous work with a 99.96 accuracy. Lokesh Karanam et al. [46] also proposed a similar IDS model, consisting of a CNN for feature selection and an LSTM for sequence analysis and classification. Thanks to the reduced dimensionality from the CNN feature selection, the computation time for the KDD CUP 99 dataset was reduced to one tenth of previous works, while maintaining a 99.6% accuracy. Another hybrid approach to intrusion detection using LSTM was proposed by Bhushan Deore et al. [47] in 2022. The author's model consisted of another CNN for feature selection, but for the detection the authors used a chimp chicken swarm optimization-based deep LSTM. The results demonstrated that the proposed model outperformed other works model using vanilla CNNs and RNNs.

In 2021, Mohammed Maithem et al. [48] proposed an IDS that resourced to a Deep Neural Network (DNN) model for intrusion detection. The authors used the KDD CUP 99 dataset and performed several preprocessing techniques in order to achieve greater performance. For encoding, the authors used one-hot encoding in all of the categorial features, such as the protocol type, and performed z-score normalization in all of the numerical features. The model topology consisted of 125 nodes at the input layer, which matches the number of features, 50 nodes in the two first hidden layers, two nodes in the third hidden layer and five nodes in the output layer, which classify packets as normal or attack, and can also provide the class of attack. The resulting accuracy of the model was very high, reaching 99.99%.

With the objective of comparing different types of learning algorithms, supervised, semi-supervised and unsupervised, Sharuka Thirimanne et al. [49] built in 2021 a model than ran the KDD CUP 99 dataset and UNSW NB15 dataset against 6 different algorithms: DNN, SVM, K-NN, One-Class SVM, K-Means and EM. The author's work concluded that supervised and semi-supervised algorithms outperformed unsupervised learning algorithms in both datasets. The authors also concluded that the DNN

outperformed the other algorithms as well. Although most works focus on supervised learning algorithms, Guo Pu et al. [50] published in 2021 a framework for intrusion detection using a hybrid unsupervised clustering-based method. The author's work combined a sub-space clustering (SSC) and one class support vector machine (OCSVM) in order to detect attacks without any previous knowledge of them, and then evaluated the method against the KDD CUP 99 dataset. In the paper, the authors concluded that while their method outperformed some other known methods in their literature, future work should focus on developing an effective feature selection method and parallelizing the unsupervised learning method with another in order to achieve better results. In 2022, Lakshmanarao et al. [51] researched different combinations of feature selection techniques applied to different ML and DL algorithms in order to select the best possible model to detect intrusions. For feature selection, the authors compared SelectKBest feature selection, impurity-based feature selection and mutual information-based feature selection techniques. For the ML and DL algorithms, the authors compared LR, KNN and DT. Combining all three feature selection techniques, the author achieved a 98% accuracy rate on the CICIDS2017 dataset. Yu Yan et al. [52] also researched different DL methods for feature extraction and compared them using an LSTM for classification against the UNSW NB15 dataset, obtaining 92% accuracy.

Focusing on a specific kind of network environment for intrusion detection, Ting Ma et al. [53] proposed in 2021 a lightweight IDS for fifth generation access networks (5G) intrusion detection by exploring physical layer attributes, since physical layer attributes are mostly unique and difficult to impersonate. For this, the authors developed a model using a LSTM network as a classification mechanism to distinguish different transmitters based on their channel state information features, and after identifying the different transmitters, the traffic is processed by a second LSTM network in order to identify malicious attacks. The end results of the proposed IDS demonstrate an improved detection accuracy against previous works and that it can successfully prevent spoofing attacks.

In another specific kind of network environment, Jisna P et al. [54] affirmed that cloud environments are evermore used, thanks to their flexibility and scalability for massive distribution of content and resourced, and as such, IDSs should be adapted to be able to identity malicious traffic in these environments. The authors propose a cloud-based DL LSTM IDS model and evaluate it against similar works. The model consisted of a

SCAE+SVN model for feature selection and an LSTM model for feature learning and classification. Using the KDD CUP 99 for training and testing, the authors achieved a 99.6% accuracy.

There has been some research done on IDSs for industrial settings, more specifically for supervisory control and data acquisition (SCADA) networks, the network of systems in charge of industrial machinery such as the PLCs. Jun Gao et al. [55] proposed in 2021 an Omni SCADA IDS, consisting of a combined model of a FNN to detected temporally uncorrelated attacks and an LSTM for temporally correlated attacks. The combined model achieved an F1-Score of 99.68% regardless of the temporal relevance of each packet.

In the transportation industry, there has been also some research done in order to detect intrusion in the context of a singular vehicle. Although the modern car is a complicated agglomerate of technologies that use a centralized system called controller area network (CAN) in order to manage and control communications between the electronic control units (ECU), the CAN system does not typically support authentication or authorization mechanisms, being especially susceptible to intrusions and attacks. Delwar Hossain et al. [56] proposed an LSTM-based IDS to mitigate the CAN system deficiencies. In order to accomplish this, the authors developed a model consisting of an LSTM network and trained it with real data from an experimental car. After optimizing the hyperparameters, the authors achieved a 99.99% accuracy. In similar research, Tianqi Yu et al. [57] also proposed an LSTM-based IDS in order to detect intrusion from a car's ECUs to the CAN, but achieving a lower mean accuracy between the different attacks of 96%.

Jitti Abraham et al. [58] published in 2021 a review of the state of art for intrusion detection and prevention in networks using ML and DL, and concluded that while an ever-growing number of works are being published in this area, there is an absence of real-world network traffic analysis, which will impact the end result of applying these models in a real-world scenario. As our work focuses on applying a NIDS model to real-world network traffic, we're answering to this call for action.

Chapter 3 – Methodology

In accordance with our introduction and aligned with our background study, we developed a DL NIDS using a stacked-LSTM architecture. Afterwards, training the model, we run it against real world network data and record the results.

3.1. Building the NIDS

Our DL NIDS architecture is shown in Figure 1. All components of the DL NIDS were implemented as a Python application.

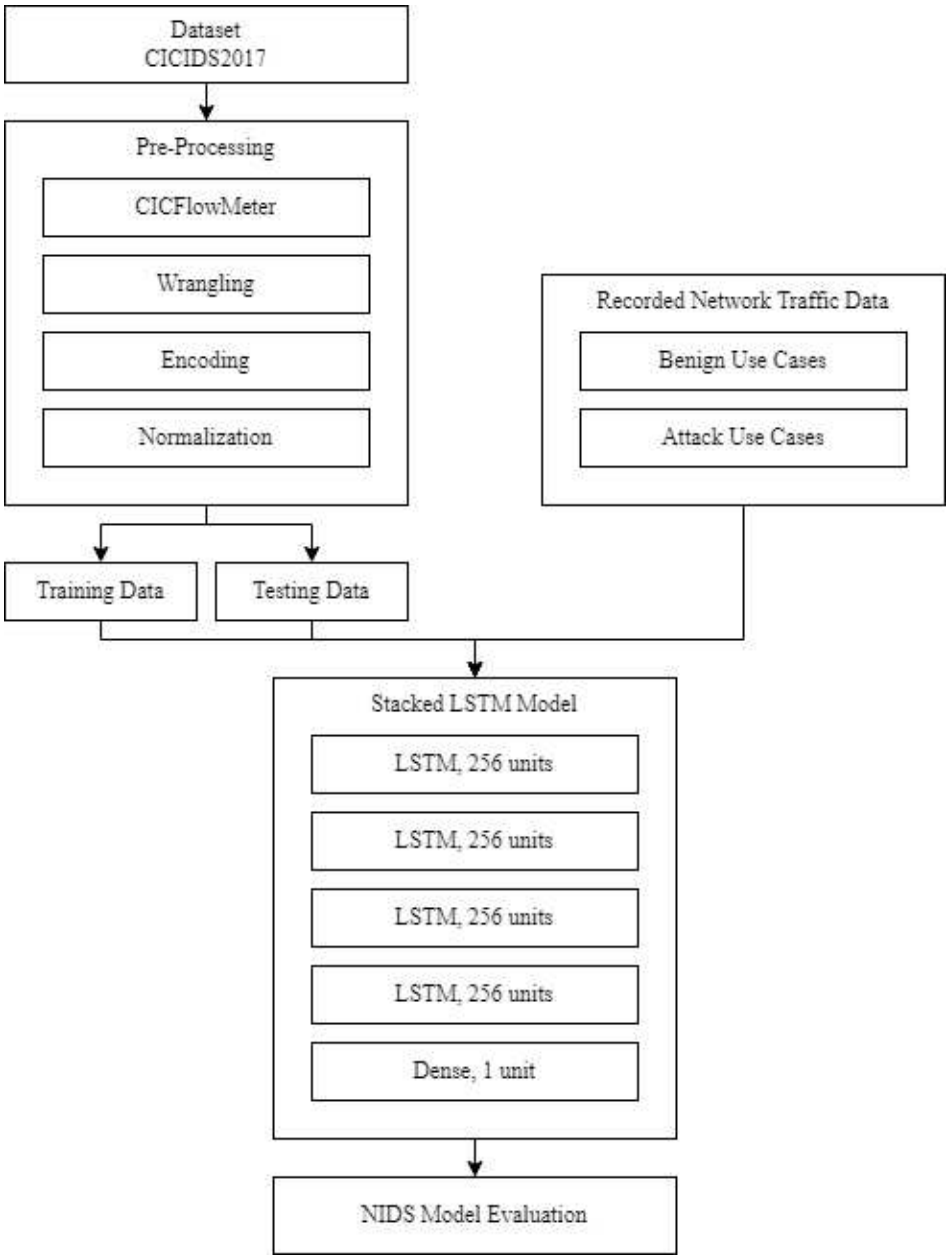


Figure 1 – Anomaly-based DL NIDS architecture

In the following sections we will further detail each aspect of our architecture.

3.1.1. Dataset

Aligned with our background study, we use the Intrusion Detection Evaluation Dataset CICIDS2017 [59] from the Canadian Institute for Cybersecurity (CIC). This dataset was recorded with the objective of resembling up-to-date attacks as since the date of its recording, most of the datasets available were recorded prior to 2000 and therefore low in resemblance to modern types of network traffic and ports used in today's network communications.

The CICIDS2017 dataset was captured from 9 AM, Monday, July 3, 2017, to 5 PM on Friday, July 7, 2017, for a total of five days.

In order to resemble a realistic network traffic, the recorded network was based on the user behavior of 25 people using systems for several kinds of applications based on HTTP, HTTPS, FTP, SSH and SMTP protocols. The attacks performed in the recording include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS, in accordance with Table 1.

Day of Recording	**Network Activity Recorded**
Monday	Benign
Tuesday	Benign, FTP-Patator, SSH-Patapor
Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS slowhttptest, DoS slorloris, Heartbleed
Thursday	Benign, Web Attack Brute Force, Web Attack SQL Injection, Web Attack XSS, Infiltration
Friday	Benign, Bot, PortScan, DDoS

Table 1 – CICIDS2017 dataset contents

The CICIDS2017 is a 15-class labelled dataset (each flow is classified with benign or has one of the 14 attack classes demonstrated in the right columns of Table 1) and has 83 features, as seen in Annex A. While a typical network traffic recording has less than 20 features, such as the ones recorded by default from typical network capture software (e.g., Wireshark), the CICIDS2017 is much more feature rich. The reasoning for this, is that the dataset's authors used a network traffic analysis tool called CICFlowMeter, which we will further detail in the next section.

For our purpose, we converted the dataset to a binary labelled dataset where each instance can either be classified as benign or as an attack. The final dataset has 2830540 instances, where 2359087 (83.34%) are classified as benign and 471453 (16.66%) are classified as an attack.

3.1.2. Pre-processing

CICFlowMeter [60] is a network traffic analyzer developed by CIC, the same organization as the authors for CICIDS2017. The tool receives as input either a PCAP file or performs analysis in realtime of a network adapter. After analyzing, the tool processes the captured network packets, converts them to flows, and outputs the results to a CSV with 83 features. This tool is crucial for our model, as it will be used for both training and testing the model, as well as for the pre-processing of the real-world network traffic captured.

One important aspect of the CICFlowMeter tool is that the resulting CSV file is not a 1-to-1 mapping to the original recorded network packets but is instead a conversion of individual network packets to network flows.

While network packets comprise of individual segments of data with a unidirectional attribute, network flows define a sequence of the network packets. With network flows, we can better understand the flow of network communications between two endpoints, resulting in a better candidate for intrusion detection.

The CICIDS2017 dataset already has available the resulting CSV files of processing each of the five recorded days of network traffic through CICFlowMeter, and as such the pre-processing step of training the model is accelerated. Although we didn't have to run CICFlowMeter, the resulting CSV files had a few encoding and whitespace inconsistencies which we had to clean.

After cleaning the inconsistencies, we merge the CSV files in a single pandas dataframe [61]. After this, we clean the rows with NaN values and the FlowID feature, as it is simply a primary key for each flow. We sort the rows by timestamp to guarantee that the network flows are ordered according to their real development and after sorting we drop the timestamp feature as it won't be needed.

Regarding the port numbers, since the dataset has both the source port and the destination port, we merged them in a single feature as seen in Algorithm 1, preserving the port that maps the flow to a single network application. While having both ports is relevant from a manual network analysis perspective, the reason these won't be useful for a ML model is that one of the ports is always a dynamic port typically attributed in the NAT process of network routing. These dynamic ports usually belong to the high range of ports – from 49152 to 65535. The lower number port belongs to a specific network application such as HTTP (port 80), HTTPS (port 443) or SSH (port 22).

```
for Row in Dataset do
  sp ← SourcePort
  dp ← DestinationPort
  if sp <= dp then
    Port = sp
  else
    Port = dp
  end if
  if Port >= 4096 then
    Port = 4096
  end if
  Row ← Row + Port
end for
```

Algorithm 1 – Port number conversion

One of the key parts of our work is regarding our choice of pre-processing on the source and destination features. Originally, these features had an IP address, which while it served as an identifier for each system in the network, it's hard to translate as a feature for a ML model. Two of the most common approaches to solve this problem are removing these features altogether or using a dictionary to translate the IP addresses to a number, which can be reversed in the end in order to identify a malicious IP address. While both techniques have been proven to achieve high accurate ratings, they're limited for model transpositions:

1. Removing the features altogether removes precious context such as the general network location and results in the incapacity of answering questions like: is the traffic coming from inside the network? Or is coming from outside? Using RDP from an internal system to another internal system is a very different situation from using RDP from outside the network to the inside.
2. Using a dictionary effectively maps individual systems and is able to detect situations such as: is all traffic coming from the same IP address? Is this individual

IP address trying to reach all of the network consequently? However, the problem is that while this is a valid method for intrusion detection in the same network context as the trained model, if we transpose this model to a different network, it will have a high rate of misclassifications as in different networks we can have very different traffic coming from the same translated IP address. Additionally, an IP address (translated or not) is effectively a categorical feature, which means that encoding it correctly would result in a dimensionally problematic dataset.

We opted to implement a trade-off between the two most common options – we converted each IP address to a binary classification of either Internal (whenever the IP address belong to a private address space) or External (whenever the IP address doesn't belong to a private address space). This can be seen in Algorithm 2. As both Internal and External are universal features of every network flow, using this method not only provides more network context than simply removing the source and destination features, but also enables an efficient transition of the model to different networks as the features are context-independent.

```
for Row in Dataset do
  if IP starts with "192.168." or "172.16." or "10." then
    IP = Internal
  else
    IP = External
  end if
end for
```

Algorithm 2 – IP address conversion

As all ML models require that each input and output variables be numeric, another key part of every successful model is the encoding and scaling of the dataset features.

Encoding data is the process of transforming some input to numbers, usually in a way that is reversible and allows the translation between the resulting output and the original input. However, we must be careful when encoding as not every feature is the same – they can either be categorical or ordinal, the difference being that while ordinal features comprise of a finite set of discrete value with a ranked ordering between values (e.g., 10 bytes is smaller than 30 bytes), categorical features have no relationship between values (e.g., port 22, port 80, port 443).

When encoding categorical features, it's not enough to convert them to a number as it can be misleading to the ML model. If we simply convert every category to a different

number, we are forcing an ordinal encoding, which ultimately results in a dataset feature with a natural ordering between categories.

In order to correctly encode the categorical features in a way that is understandable by a ML model, we performed a technique called one-hot encoding. Using one-hot encoding, we convert each possible category from a specific feature, in a new feature of its own that can have either 1 if it belongs to the category or 0 if it doesn't, as shown in Figure 2.

Protocol Type
TCP
TCP
UDP
ICMP
TCP
ICMP

→

Protocol Type TCP	Protocol Type UDP	Protocol Type ICMP
1	0	0
1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

Figure 2 – One-Hot Encoding explanation

We perform one-hot encoding for every categorical feature, which are the Protocol, the Destination, the Source, and with some custom tuning, the Port feature. As one-hot encoding highly increases the dimensionality of a dataset, we couldn't transpose every single port. As such, we opted to perform one-hot encoding for every flow with a port from 0 to 4096. For every flow after port 4096, we classified them as 4096. While this means that the NIDS won't be able to distinguish different ports over 4096, we still encompass the most frequently used ports in both benign and attack network use cases (e.g., RDP, SSH, FTP, DNS).

Another important aspect of the pre-processing is the feature scaling. Since the range of values for some of our numerical ordinal features vary widely, we performed the min-max scaling technique on all of the non-categorical features. This is commonly known as normalization of values, and scales all value to a fixed range between 0 and 1.

After encoding and scaling all features, we save the encoding and scaling configurations to a file in order to apply later the same context of encoding and scaling for the real-life network traffic data.

Lastly, we split the dataset using sci-kit's train_test_split function [62], which randomly distributes the dataset in two smaller datasets: a training dataset, which is used to fit the model, and a testing dataset, which is used for unbiased model evaluation and

tuning. We dedicate 75% of the complete dataset for the training dataset and 25% for the testing dataset.

3.1.3. Stacked LSTM Model

Following our background study, we propose the usage of a stacked LSTM model for the DL NIDS. Our final model topology is defined by the code in Figure 3.

Layer (type)	Output Shape	Parameters
LSTM	(None, 1, 256)	1248256
Dropout	(None, 1, 256)	0
LSTM	(None, 1, 256)	525312
Dropout	(None, 1, 256)	0
LSTM	(None, 1, 256)	525312
Dropout	(None, 1, 256)	0
LSTM	(None, 256)	525312
Dropout	(None, 256)	0
Dense	(None, 1)	257
Activation	(None, 1)	0

Figure 3 – Stacked LSTM model topology

The LSTM is a subset of the RNN family of models and is best suited for the classification of temporal-dependent data, such as the data from a network intrusion problem. An LSTM unit consists of a cell, an input gate, an output gate and a forget gate.

One of the biggest challenges of training a large model is the overfitting of the training data. A lot of times this results in the model learning statistical noise as a classification feature and, when applied to different environments, can result in poor performance. As such, we use the dropout technique as a preventive measure to reduce this phenomenon.

The dropout technique is a regularization method that randomly drops nodes from the input and the hidden layers in the model. By dropping these nodes, the model has an easier job of filtering out the statistical noise and learning the defining characteristics of what it is trying to learn.

In our model, we implement a dropout layer after each of the LSTM's layers with a rate of 0.1, which means that the 10% of the input units will be dropped for each LSTM layer.

The loss function of a model is the function that computes the distance between the current output of the algorithm and the desired one. In order to have the best possible model, we want this distance to be as small as possible, which means trying to get the

output of the function as close as possible to zero. There are many different possible loss functions to use in a model, but as our problem is a binary classification one, we use the cross-entropy function, also known as the log loss function.

While the loss function let us know how close the model is to the desired output, in order to optimize the learning process of the model we use an optimization algorithm. These algorithms work by finding the best possible value for the weights that best minimize the error when mapping inputs to outputs, and then update them in the model. For our purposes we use the Adam optimization algorithm [63] from Diederik Kingma et al., which is an extension to the stochastic gradient descent and has been seeing great adoption since its inception in 2015. The main difference between the Adam algorithm and the classical stochastic gradient descent is that while the latter maintain a single learning rate for all weight updates, Adam can dynamically update it using adaptive gradient algorithms and the root mean square propagation.

At the end of a model, in order to define how the weighted sum of the previous layers results in a prediction, we must use an activation layer. This layer computes the result of an activation function, which needs to be chosen and adapted in regard to the problem in hand. As what we are trying to accomplish is a binary classification problem, the recommended activation function is a sigmoid activation function.

As for the training of the model, we run the fit function for 25 epochs with a batch size of 128.

3.2. Running real-world data through the NIDS

After the training of the model, the next part of our work aims to evaluate if our NIDS is capable of accurately identifying intrusions when confronted with real-world data from outside of the training and testing dataset.

In order for a NIDS to evaluate the network traffic data and identify possible intrusions, it needs to the network flows in some way. In a real network environment, most NIDSs receive them via an integration with either a firewall or a network sniffing tool, but for academic purposes most authors opt to use pre-recorded network traffic data, usually in the format of a PCAP file. In our work we had the opportunity to choose either of them thanks to the CICFlowMeter tool, because while the main purpose of the tool is to extract features from a network traffic data in a PCAP format, the tool's authors also

included the capability of extracting these features in realtime by sniffing the network data of a chosen network adapter. Additionally, the tool is open-source and its code is publicly available in GitHub, which means that in the future we can link these two pieces directly.

3.2.1. Real-world network traffic data recording

As one of our objectives is to evaluate the model against real-world network traffic data, we need to define:

1. What do we want to record.
2. Where do we want to record.
3. How do we want to record.

In regard to what do we want to record, we opted to record several different benign and malicious network use cases, as described in the Table 2. These use cases range from pure white noise network traffic (as to establish a baseline of the network traffic data) to synthetic network attacks. By analyzing the classification given by the NIDS to all of these different use cases, we will be able to understand if the NIDS can accurately predict if there's a network intrusion ongoing or not.

Use case	**Type of use case**	**Number of packets**	**Number of flows**	**Description**
White noise	Benign	1069	108	Background network traffic from the recorded system with the default background services from Windows 10
Internet browsing	Benign	66688	322	General internet browsing usage consisting of YouTube, Google Spreadsheets and Google Search
WhatsApp	Benign	1020	64	Message and images exchange via the WhatsApp Windows desktop client
Spotify	Benign	36189	373	Music streaming via the Spotify Windows desktop client
C2	Attack	111	23	DNS and IP traffic to a random list of known C2 destinations

Use case	Type of use case	Number of packets	Number of flows	Description
DGA	Attack	398	37	Domain generation algorithm traffic using random labels and top-level domains
Imposter	Attack	62	12	DNS traffic to a list of imposter domains
Miner	Attack	118	22	Stratum mining protocol traffic to known cryptomining pools
Scan	Attack	263	13	Port scan of random RFC 5737 addresses using common TCP ports
Sink	Attack	131	26	Connection to known sinkholed destinations run by security researchers
Spambot	Attack	173	47	Resolve and connection to random Internet SMTP servers to simulate a spam bot
Tunnel DNS	Attack	216	62	DNS tunneling requests to *.sandbox.alphasoc.xyz
Tunnel ICMP	Attack	209	11	ICMP tunneling traffic to an Internet service operated by AlphaSOC

Table 2 – Recorded network use cases

While the benign use cases were recorded by performing the use cases themselves, all of the malicious use cases were generated with the help of a malicious network traffic generator tool called flightsim [64], which is an open-source tool made by the company Alphasoc that is capable of generating several different kinds of synthetic network attacks in order to evaluate if security teams are capable of detecting and responding to them.

As to where we want to record, we chose to capture the network traffic in a desktop system inside of a controlled network environment. While we could use a more complex capturing scenario such as using port mirroring in a switch, with our pre-processing the results should still be accurate and proportionally similar to using the model in a bigger network.

Finally, in regard to how we want to record, we use the widely popular network capture tool Wireshark [65]. We captured all traffic from the network connected adapter without any filter while running all of the use cases and exported the results to a PCAP file.

3.2.2. Feeding the captured data to the NIDS

With the resulting PCAP file, we now have the raw data that will be fed to our DL NIDS. But in order to respect the DL model, we need to use the same input shape as the one used for training. As such, we run all of the PCAP files through the CICFlowMeter tool in order to extract the new features and export them to a CSV file.

Afterwards we need to run the same pre-processing techniques such as converting internal IP addresses to just internal, dropping the unnecessary timestamp feature, using the same scaling factor and weights from the training phase, and encoding the categorical features. In order to perform all of the same pre-processing methodology, we reuse the same Python code we used for the training of the model.

At the end, we import the saved model, run the predict function against the pre-processed data, and export the results to a CSV file in order to evaluate the results.

Chapter 4 – Results and Discussion

Aligned with our objectives, we performed two different experiments – the first one is the evaluation of the performance of our NIDS using our novel pre-processing techniques against the CICIDS2017, and the second one is the evaluation of our NIDS against real-world network traffic data.

4.1. CICIDS2017 evaluation

For the first experiment, we measure our results with the evaluation matrics of accuracy, precision, recall and F1-Score. These evaluation metrics depend on the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) and can be mapped in a confusion matrix.

The accuracy, which is defined as the total correctly classified samples divided by the total number of samples, is given by the equation X, where TP equals True-Positives, TN equals true negatives, FP equals false positives and FN equals false negatives. The equation for accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

The precision, which is defined as the actual correction predictions divided by the total number of positively classified samples, is given by following the equation:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

The recall, which is also known as the sensitivity or true positive rate and is defined as the numbers of true positives divided by the total number of true positives and false negatives, is given the equation:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

The F1-Score, which a harmonic average of the precision and recall metrics and is typically known as the more useful metric in order to evaluate a model, is given by the equation:

$$\text{F1} - \text{Score} = 2.\frac{Precision.Recall}{Precision + Recall} \quad (4)$$

Using the CICIDS2017 dataset for training and testing of our model, we obtained the confusion matrix in Figure 4.

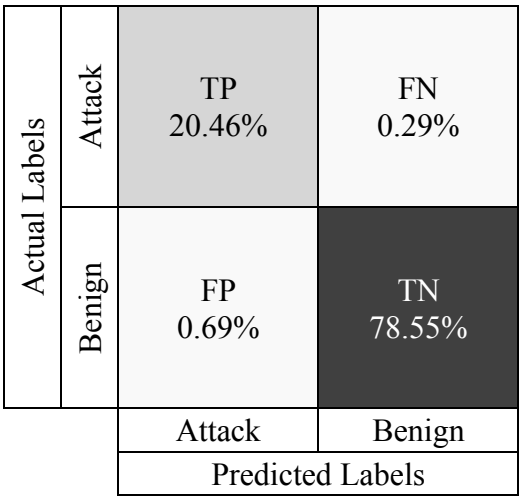


Figure 4 – Confusion matrix with our results

Transposing the results to the defined evaluation metrics, we get the results shown in Table 3.

	Precision	Recall	F1-Score
Attack	0.9671	0.9858	0.9764
Benign	0.9963	0.9912	0.9937

Table 3 – Evaluation metrics

These metrics reveal that our DL NIDS is capable of accurate classification of network flows as either attack or benign. In fact, the F1-Score for attack classification is 97.64% and the F1-Score for benign classification is 99.37%. The accuracy of the model is 99.01%, which is above most related works that used the CICIDS2017 dataset, as seen in Table 4.

Authors	**Model architecture**	**Accuracy**
Figueiredo et al. (this study)	Stacked LSTM	99.01%
Kim et al. [66]	CNN-LSTM	98.07%
Zhang et al. [43]	Single LSTM layer	97.79%
Alsyaibani et al. [32]	Bidirectional LSTM	97.72%
Ferrag et al. [67]	Unspecified	97.28%
Nayyar et al. [68]	Single LSTM layer	96.70%
Alin et al. [69]	CNN	92.00%

Table 4 – Comparison with similar studies

4.2. Real-world network traffic data evaluation

To evaluate our NIDS efficacy with real-world network traffic data, we can't use the same evaluation metrics as the ones used for the CICIDS2017 dataset, as real-world data isn't labelled. As such, we look at the number of network flows classified as malicious in each of the recorded use cases. The results are further detailed in Table 5.

Use Case	# of Benign classifications	# of Attack classifications	Percentage of Attack classifications
White noise	107	1	0.93%
Internet browsing	320	2	0.62%
WhatsApp	64	0	0.00%
Spotify	372	1	0.27%
C2	23	0	0.00%
DGA	37	0	0.00%
Imposter	12	0	0.00%
Miner	22	0	0.00%
Scan	12	1	7.69%
Sink	26	0	0.00%
Spambot	40	7	14.89%
Tunnel DNS	61	1	1.61%
Tunnel ICMP	11	0	0.00%

Table 5 – Real-world network data captured

While these results were subpar in comparison with the results from CICIDS2017 dataset, as written in our introduction and methodology, a DL NIDS focuses not on IOCs but on heuristics and rules.

A typical network intrusion is not achieved via a single network flow, and as such, it is not expected to immediately return a binary classification based on the analysis of several network flows – in order to be properly implemented in a production environment, a DL NIDS must output, at any given time, a probability that an intrusion might be occurring.

The best approach for this, is using a ratio of benign-classified network flows and attack-classified network flows. We can see these results in the last column of Table 6.

Starting with the benign use cases, we can observe that less than 1% of network flows were classified as an attack. This is a good result, as we can define the threshold of our NIDS as 1% and assume that below that value, we won't be under a network intrusion.

As for the attack use cases, we can observe that while for most of the attacks the NIDS outputted a 0% chance of being under a network intrusion, for the Scan, Spambot and

Tunnel DNS attack the results were over 1%, with Spambot outputting almost 15%. We can interpret these results in two ways:

1. Our NIDS can detect some categories of attacks, as the resulting output of some of the categories of attack are clearly over the 1% benchmark established by comparing the benign use cases.

2. Our NIDS can't detect other categories of attacks since it outputted 0% in most of them.

We believe that the reason the NIDS didn't perform as well with the real-word data as with the CICIDS2017 is mainly related to the recorded use cases of both datasets. While both datasets have several different types of network intrusions use cases, they don't overlap on most of them. As such, we trained our model to detect the specific network intrusion attacks used in CICIDS2017, but as our real-world dataset was recorded with different kinds of network intrusion attacks, the model isn't able to detect them accurately.

The exception to this rule, is the similar attacks present in both datasets – more specifically, the denial-of-service attacks. As seen in Table 6, our NIDS accurately detected intrusions in the two recorded use cases that are similar to a denial of service.

Chapter 5 – Conclusions and Future Works

In our work, we reviewed the current state of the art in anomaly-based intrusion detection systems supported by deep learning architectures. We conclude that since its first iteration in 1980, there has been a lot of research in advancing its heuristics capabilities. In the last decade, with the uprising of machine learning and deep learning techniques, which was enabled by newer and more powerful hardware, we have seen research done to improve these models and reduce the false positive rates. Some of the recent works even reach over 90% accuracy rate, which capacitates them for production usage, although arguably for the purpose of detection and not automated prevention.

We propose a novel pre-processing technique that can be used in order to obtain a high-accuracy trained anomaly-based DL NIDS. We pair our pre-processing technique with a stacked LSTM architecture, as it is indicated for temporal context-related problems such as intrusion detections.

Using our DL NIDS model with the CICIDS2017 dataset, we obtained a 99.01% accuracy, which makes it the most accurate model using this specific dataset according to our literature review.

In regard to using our DL NIDS model with real-world network data, we find that it can detect the network intrusion use cases that were used for the original training.

In our work we had the limitations of using a smaller network environment than one used in a typical corporate network. We also used synthetically generated network intrusion attacks instead of performing real network intrusion attacks, which did contribute to bigger false negative rates. Another limitation we faced is the port number encoding, as while the usage of one-hot encoding will contribute to an even more accuracate model, it is susceptible to dimensionality problems and will need a very high amount of system memory.

For the research question “how accurate is a DL model in intrusion detection?”, we conclude that with the usage of our model, we were able to reach 99.01% accuracy, which is above all of the previous works that used the CICIDS2017 dataset. This is an accurate enough model to consider adding automated prevention capabilities without disturbing the well-being of the network in which its implemented.

As for the research question “how can we transpose a simulated dataset-trained DL NIDS to a real word network environment?”, we conclude that with the usage of our pre-processing methodology, we can transpose a labelled dataset trained DL NIDS model to a different network and be able to detect similar intrusion use cases as the ones used for training. As our DL NIDS was only capable to detect these previously known network intrusion use cases, its not suitable for a fire-and-forget implementation in a real network as the sole intrusion detection mechanism. In order to improve our model, future works should consider training the model with more network intrusion use cases using an existing framework of attacks, such as the MITRE ATT&CK. As the DL NIDS is able to learn the patterns of network intrusion use cases, its not entirely relevant the variant of the attack performed for the training, as the model is capable of learning the inherent behaviour of each type of network intrusion use case.

We recommend that future works consider training the model using a multi-class approach, as it might improve even further the accuracy and will better adapt in future automated intrusion blocking mechanisms. We also recommend that future works consider the usage of IPv6 and its attributes for intrusion detection, as while its still not widely used in internal corporate networks, it might be relevant in the future.

Lastly, the contents of this thesis were adapted to an article and submitted to MDPI’s special issue “Network Intrusion Detection Using Deep Learning”, currently waiting for review and approval.

References

[1] INTERPOL, “INTERPOL Report.” https://www.interpol.int/en/News-and-Events/News/2022/Financial-and-cybercrimes-top-global-police-concerns-says-new-INTERPOL-report.

[2] M. & Company, “Cybersecurity trends: Looking over the horizon.” https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/cybersecurity-trends-looking-over-the-horizon.

[3] J. P. Anderson, “Computer security threat monitoring and surveillance,” *Tech. Rep. James P Anderson Co Fort Washingt. Pa*, p. 56, 1980, doi: citeulike-article-id:592588.

[4] D. E. Denning, “An intrusion-detection model,” in *Proceedings - IEEE Symposium on Security and Privacy*, 2012, no. 1986, pp. 118–131, doi: 10.1109/SP.1986.10010.

[5] A. Jallad, J. B. Data, K. Al Jallad, M. Aljnidi, and M. S. Desouki, “Anomaly detection optimization using big data and deep learning to reduce false - positive,” *J. Big Data*, pp. 1–12, 2020, doi: 10.1186/s40537-020-00346-1.

[6] J. Bloomberg, “The cybersecurity risk that dwarfs all others,” *Forbes*.

[7] Y. Roumani, “Patching zero-day vulnerabilities: an empirical analysis,” *J. Cybersecurity*, vol. 7, no. 1, p. tyab023, Jan. 2021, doi: 10.1093/cybsec/tyab023.

[8] Y. Wang, J. Wong, and A. Miner, “Anomaly intrusion detection using one class SVM,” *Proc. fron Fifth Annu. IEEE Syst. Man Cybern. Inf. Assur. Work. SMC*, pp. 358–364, 2004, doi: 10.1109/iaw.2004.1437839.

[9] J. Shum and H. A. Malki, “Network intrusion detection system using neural networks,” *Proc. - 4th Int. Conf. Nat. Comput. ICNC 2008*, vol. 5, pp. 242–246, 2008, doi: 10.1109/ICNC.2008.900.

[10] X. Fang and L. Liu, “Integrating artificial intelligence into Snort IDS,” *2011 3rd Int. Work. Intell. Syst. Appl. ISA 2011 - Proc.*, 2011, doi: 10.1109/ISA.2011.5873435.

[11] C. F. Tsai and C. Y. Lin, “A triangle area based nearest neighbors approach to intrusion detection,” *Pattern Recognit.*, vol. 43, no. 1, pp. 222–229, Jan. 2010,

doi: 10.1016/J.PATCOG.2009.05.017.

[12] T. Mehmood and H. B. M. Rais, “Machine learning algorithms in context of intrusion detection,” *2016 3rd Int. Conf. Comput. Inf. Sci. ICCOINS 2016 - Proc.*, pp. 369–373, 2016, doi: 10.1109/ICCOINS.2016.7783243.

[13] Z. He, T. Zhang, and R. B. Lee, “Machine Learning Based DDoS Attack Detection from Source Side in Cloud,” *Proc. - 4th IEEE Int. Conf. Cyber Secur. Cloud Comput. CSCloud 2017 3rd IEEE Int. Conf. Scalable Smart Cloud, SSC 2017*, pp. 114–120, 2017, doi: 10.1109/CSCloud.2017.58.

[14] R. Dechter, “Learning While Searching in Constraint-Satisfaction-Problems.,” *Aaai*, no. May, pp. 178–185, 1986.

[15] “NVIDIA CEO bets big on deep learning,” [Online]. Available: https://venturebeat.com/business/nvidia-ceo-bets-big-on-deep-learning-and-vr/.

[16] J. Esmaily, R. Moradinezhad, and J. Ghasemi, “Intrusion detection system based on Multi-Layer Perceptron Neural Networks and Decision Tree,” *2015 7th Conf. Inf. Knowl. Technol. IKT 2015*, pp. 1–5, 2015, doi: 10.1109/IKT.2015.7288736.

[17] N. Lyamin, D. Kleyko, Q. Delooz, and A. Vinel, “AI-Based Malicious Network Traffic Detection in VANETs,” *IEEE Netw.*, vol. 32, no. 6, pp. 15–21, 2018, doi: 10.1109/MNET.2018.1800074.

[18] Kunal and M. Dua, “Machine Learning Approach to IDS: A Comprehensive Review,” *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, pp. 117–121, 2019, doi: 10.1109/ICECA.2019.8822120.

[19] E. D. Alalade, “Intrusion Detection System in Smart Home Network Using Artificial Immune System and Extreme Learning Machine Hybrid Approach,” *IEEE World Forum Internet Things, WF-IoT 2020 - Symp. Proc.*, pp. 20–21, 2020, doi: 10.1109/WF-IoT48130.2020.9221151.

[20] I. Sofi, A. Mahajan, and V. Mansotra, “Machine Learning Techniques used for the Detection and Analysis of Modern Types of DDoS Attacks,” *Int. Res. J. Eng. Technol.*, vol. 4, no. 6, pp. 1085–1092, 2017, [Online]. Available: https://irjet.net/archives/V4/i6/IRJET-V4I6200.pdf.

[21] M. J. R. Dennis and X. Li, “Machine-Learning and Statistical Methods for DDoS Attack Detection and Defense System in Software Defined Networks,” 2018,

[Online]. Available: https://digital.library.ryerson.ca/islandora/object/RULA:7564/datastream/OBJ/download/Machine-learning_and_statistical_methods_for_DDoS_attack_detection_and_defense_system_in_software_defined_networks.pdf.

[22] L. Yang and H. Zhao, "DDoS attack identification and defense using SDN based on machine learning method," *Proc. - 2018 15th Int. Symp. Pervasive Syst. Algorithms Networks, I-SPAN 2018*, pp. 174–178, 2019, doi: 10.1109/I-SPAN.2018.00036.

[23] A. O. Alzahrani and M. J. F. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Futur. Internet*, vol. 13, no. 5, 2021, doi: 10.3390/fi13050111.

[24] P. Rivas *et al.*, "Machine Learning for DDoS Attack Classification Using Hive Plots," *2019 IEEE 10th Annu. Ubiquitous Comput. Electron. Mob. Commun. Conf. UEMCON 2019*, pp. 0401–0407, 2019, doi: 10.1109/UEMCON47517.2019.8993021.

[25] J. Zhang, Q. Liang, R. Jiang, and X. Li, "A Cognitive Approach for Multiple Attack Vectors Based DDoS with GDBT algorithm," 2019.

[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[27] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 3, pp. 2048–2057, 2015.

[28] Zander Danko, "Neon prescription... or rather, New transcription for Google Voice," [Online]. Available: https://blog.google/products/google-voice/neon-prescription-or-rather-new/.

[29] M. Stampar and K. Fertalj, "Artificial intelligence in network intrusion detection," *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, no. June, pp. 1318–1323, 2015, doi: 10.1109/MIPRO.2015.7160479.

[30] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory

Networks for Anomaly Detection in Time Series," *ESANN 2012 proceedings, 20th Eur. Symp. Artif. Neural Networks, Comput. Intell. Mach. Learn.*, no. April, pp. 1–650, 2012.

[31] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential LSTM Neural Networks autoencoders," *26th IEEE Signal Process. Commun. Appl. Conf. SIU 2018*, pp. 1–4, 2018, doi: 10.1109/SIU.2018.8404689.

[32] O. M. A. Alsyaibani, E. Utami, and A. D. Hartanto, "An Intrusion Detection System Model Based on Bidirectional LSTM," *3rd Int. Conf. Cybern. Intell. Syst. ICORIS 2021*, 2021, doi: 10.1109/ICORIS52787.2021.9649612.

[33] M. D. Rokade and Y. K. Sharma, "MLIDS: A machine learning approach for intrusion detection for real time network dataset," *2021 Int. Conf. Emerg. Smart Comput. Informatics, ESCI 2021*, pp. 533–536, 2021, doi: 10.1109/ESCI50559.2021.9396829.

[34] S. A. Althubiti, E. M. Jones, and K. Roy, "LSTM for Anomaly-Based Network Intrusion Detection," *2018 28th Int. Telecommun. Networks Appl. Conf. ITNAC 2018*, pp. 20–22, 2019, doi: 10.1109/ATNAC.2018.8615300.

[35] J. Meira, "Comparative Results with Unsupervised Techniques in Cyber Attack Novelty Detection," *Proceedings*, vol. 2, no. 18, p. 1191, 2018, doi: 10.3390/proceedings2181191.

[36] T. T. H. Le, Y. Kim, and H. Kim, "Network intrusion detection based on novel feature selection model and various recurrent neural networks," *Appl. Sci.*, vol. 9, no. 7, 2019, doi: 10.3390/app9071392.

[37] S. S. Swarna Sugi and S. R. Ratna, "Investigation of machine learning techniques in intrusion detection system for IoT network," *Proc. 3rd Int. Conf. Intell. Sustain. Syst. ICISS 2020*, pp. 1164–1167, 2020, doi: 10.1109/ICISS49785.2020.9315900.

[38] S. H. Park, H. J. Park, and Y. J. Choi, "RNN-based Prediction for Network Intrusion Detection," *2020 Int. Conf. Artif. Intell. Inf. Commun. ICAIIC 2020*, pp. 572–574, 2020, doi: 10.1109/ICAIIC48513.2020.9065249.

[39] J. Jose, D. V. Jose, K. S. Rao, and J. Janz, "Impact of Machine Learning Algorithms in Intrusion Detection Systems for Internet of Things," *10th Int.*

Conf. Adv. Comput. Commun. ICACC 2021, pp. 0–5, 2021, doi: 10.1109/ICACC-202152719.2021.9708404.

[40] G. Abdelmoumin, D. B. Rawat, and A. Rahman, "On the Performance of Machine Learning Models for Anomaly-Based Intelligent Intrusion Detection Systems for the Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4280–4290, 2022, doi: 10.1109/JIOT.2021.3103829.

[41] K. Saurabh *et al.*, "LBDMIDS: LSTM Based Deep Learning Model for Intrusion Detection Systems for IoT Networks," *2022 IEEE World AI IoT Congr. AIIoT 2022*, pp. 753–759, 2022, doi: 10.1109/AIIoT54504.2022.9817245.

[42] S. W. Azumah, N. Elsayed, V. Adewopo, Z. S. Zaghloul, and C. Li, "A Deep LSTM based Approach for Intrusion Detection IoT Devices Network in Smart Home," *7th IEEE World Forum Internet Things, WF-IoT 2021*, pp. 836–841, 2021, doi: 10.1109/WF-IoT51360.2021.9596033.

[43] L. Zhang, H. Yan, and Q. Zhu, "An Improved LSTM Network Intrusion Detection Method," *2020 IEEE 6th Int. Conf. Comput. Commun. ICCC 2020*, pp. 1765–1769, 2020, doi: 10.1109/ICCC51575.2020.9344911.

[44] A. Dey, "Deep IDS : A deep learning approach for Intrusion detection based on IDS 2018," *2020 2nd Int. Conf. Sustain. Technol. Ind. 4.0, STI 2020*, vol. 0, pp. 19–20, 2020, doi: 10.1109/STI50764.2020.9350411.

[45] S. Amutha, R. Kavitha, S. Srinivasan, and M. Kavitha, "Secure network intrusion detection system using NID-RNN based Deep Learning," *Proc. - IEEE Int. Conf. Adv. Comput. Commun. Appl. Informatics, ACCAI 2022*, pp. 1–5, 2022, doi: 10.1109/ACCAI53970.2022.9752526.

[46] L. Karanam, K. K. Pattanaik, and R. Aldmour, "Intrusion Detection Mechanism for Large Scale Networks using CNN-LSTM," *Proc. - Int. Conf. Dev. eSystems Eng. DeSE*, vol. 2020-Decem, pp. 323–328, 2020, doi: 10.1109/DeSE51703.2020.9450732.

[47] B. Deore and S. Bhosale, "Hybrid Optimization Enabled Robust CNN-LSTM Technique for Network Intrusion Detection," *IEEE Access*, vol. 10, pp. 65611–65622, 2022, doi: 10.1109/ACCESS.2022.3183213.

[48] M. Maithem and G. A. Al-Sultany, "Network intrusion detection system using

deep neural networks," *J. Phys. Conf. Ser.*, vol. 1804, no. 1, 2021, doi: 10.1088/1742-6596/1804/1/012138.

[49] S. Thirimanne, L. Jayawardana, P. Liyanaarachchi, and L. Yasakethu, "Comparative Algorithm Analysis for Machine Learning Based Intrusion Detection System," *2021 10th Int. Conf. Inf. Autom. Sustain. ICIAfS 2021*, pp. 191–196, 2021, doi: 10.1109/ICIAfS52090.2021.9605814.

[50] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 146–153, 2021, doi: 10.26599/TST.2019.9010051.

[51] A. Lakshmanarao, A. Srisaila, and T. S. Ravi Kiran, "Machine Learning and Deep Learning framework with Feature Selection for Intrusion Detection," *2022 Int. Conf. Commun. Comput. Internet Things, IC3IoT 2022 - Proc.*, pp. 0–4, 2022, doi: 10.1109/IC3IOT53935.2022.9767727.

[52] Y. Yan, L. Qi, J. Wang, Y. Lin, and L. Chen, "A Network Intrusion Detection Method Based on Stacked Autoencoder and LSTM," *IEEE Int. Conf. Commun.*, vol. 2020-June, 2020, doi: 10.1109/ICC40277.2020.9149384.

[53] T. Ma, F. Hu, and M. Ma, "A LSTM-based channel fingerprinting method for intrusion detection," *2021 IEEE 5th Int. Conf. Cryptogr. Secur. Privacy, CSP 2021*, pp. 113–116, 2021, doi: 10.1109/CSP51677.2021.9357494.

[54] P. Jisna, T. Jarin, and P. N. Praveen, "Advanced Intrusion Detection Using Deep Learning-LSTM Network on Cloud Environment," *Proc. 4th Int. Conf. Microelectron. Signals Syst. ICMSS 2021*, pp. 0–5, 2021, doi: 10.1109/ICMSS53060.2021.9673607.

[55] J. Gao *et al.*, "Omni SCADA Intrusion Detection Using Deep Learning Algorithms," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 951–961, 2021, doi: 10.1109/JIOT.2020.3009180.

[56] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185489–185502, 2020, doi: 10.1109/ACCESS.2020.3029307.

[57] T. Yu, G. Hua, H. Wang, J. Yang, and J. Hu, "Federated-LSTM based Network Intrusion Detection Method for Intelligent Connected Vehicles," *IEEE Int. Conf.*

Commun., vol. 2022-May, pp. 4324–4329, 2022, doi: 10.1109/ICC45855.2022.9838655.

[58] J. A. Abraham and V. R. Bindu, "Intrusion Detection and Prevention in Networks Using Machine Learning and Deep Learning Approaches: A Review," *2021 Int. Conf. Adv. Electr. Electron. Commun. Comput. Autom. ICAECA 2021*, pp. 2–5, 2021, doi: 10.1109/ICAECA52838.2021.9675595.

[59] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSP 2018 - Proc. 4th Int. Conf. Inf. Syst. Secur. Priv.*, vol. 2018-Janua, no. Cic, pp. 108–116, 2018, doi: 10.5220/0006639801080116.

[60] CIC, "CICFlowMeter." https://www.unb.ca/cic/research/applications.html#CICFlowMeter.

[61] "Pandas Dataframe." https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

[62] S. Learn, "train_test_split." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[63] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014, doi: 10.48550/arxiv.1412.6980.

[64] "Flightsim," [Online]. Available: https://github.com/alphasoc/flightsim.

[65] "Wireshark," [Online]. Available: https://www.wireshark.org/.

[66] A. Kim, M. Park, and D. H. Lee, "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," *IEEE Access*, vol. 8, pp. 70245–70261, 2020, doi: 10.1109/ACCESS.2020.2986882.

[67] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, pp. 1–21, 2020, doi: 10.1016/j.jisa.2019.102419.

[68] S. Nayyar, S. Arora, and M. Singh, "Detection System," *Comput. Sci. Commun. Dict.*, pp. 393–393, 2000, doi: 10.1007/1-4020-0613-6_4827.

[69] F. Alin, A. Chemchem, F. Nolot, and O. Flauzac, "Towards a Hierarchical Deep Learning," no. April, 2020, doi: 10.1007/978-3-030-45778-5.

Annex A – CICIDS2017 extracted features

Feature Name	Description
Source IP	Source IP address
Source Port	Source port number
Destination IP	Destination IP address
Destination Port	Destination port number
Protocol	IP protocol number found in the IPv4 header
Timestamp	Timestamp of the flow
Flow duration	Duration of the flow in Microsecond
total Fwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Byte/s	Number of flow packets per second
Flow Packets/s	Number of flow bytes per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction

Feature Name	Description
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a packet
Max Packet Length	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWE
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
AVG Bwd Segment Size	Average number of bytes bulk rate in the backward direction
Fwd Header Length	Length of the forward packet header
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd AVG Packet/Bulk	Average number of packets bulk rate in the forward direction
Fwd AVG Bulk Rate	Average number of bulk rate in the forward direction

Feature Name	Description
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd AVG Packet/Bulk	Average number of packets bulk rate in the backward direction
Bwd AVG Bulk Rate	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming actives