DL-based Defense Against Polymorphic Network Attacks

by

Ulya Sabeel

A thesis submitted to the School of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Faculty of Business and Information Technology

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

January 2024

Thesis Examination Information

Submitted by: **Ulya Sabeel**

Doctor of Philosophy in **Computer Science**

Thesis title: **'DL-based Defense Against Polymorphic Network Attacks'**

An oral defense of this thesis took place on January 23, 2024 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee:	Dr. Richard Pazzi
Research Supervisor:	Dr. Shahram Shah Heydari
Research Co-supervisor:	Dr. Khalil El-Khatib
Examining Committee Member:	Dr. Khalid Elgazzar
Examining Committee Member:	Dr. Patrick Hung
University Examiner:	Dr. Qusay Mahmoud
External Examiner:	Dr. Ali Dehghantanha, University of Guelph

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Network security is of vital importance in our world dominated by internet systems. These systems are vulnerable to large-scale rapidly evolving attacks by sophisticated cyber attackers who can have an upper edge over the defensive systems. Artificial Intelligence (AI) based intrusion detection systems provide effective defense mechanisms against cyber attacks. However, these techniques often rely on the same dataset for training and validation as well as evaluation of AI models. Current research [1] also confirms that such trained models can accurately identify known/typical network attacks but perform poorly when faced with continuously evolving atypical/ polymorphic cyberattacks. Therefore, it is crucial to develop and train an AI-based Intrusion Detection System (IDS) that proactively learns to resist infiltration by such dynamically changing attacks.

For this purpose, in this research work, we propose an AI-based IDS system that can monitor and detect polymorphic network attacks with high confidence levels. We propose a novel hybrid adversarial model that leverages the best characteristics of a Conditional Variational Autoencoder (CVAE) and a Generative Adversarial Network (GAN). Our system generates adversarial polymorphic attacks against the IDS to examine its performance and incrementally retrains it to strengthen its detection of new attacks, specifically for minority attack samples in the input data. The employed attack quality analysis ensures that the adversarial atypical/polymorphic

attacks generated through our system resemble realistic network attacks. Our experiments showcase the exceptional performance of the proposed IDS by training it using the CICIDS2017 and CICIoT2023 benchmark datasets and evaluating its performance against several atypical/polymorphic attack flows. The results indicate that the proposed technique, through adaptive training, learns the pattern of dynamically changing atypical/polymorphic attacks and identifies such attacks with high IDS proficiency. Additionally, our IDS surpasses various state-of-the-art anomaly detection and class balancing techniques.

Keywords: Attack quality; Atypical/Polymorphic attacks; Deep learning; Feature Profile; Intrusion Detection System

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Ulya Sabeel

Statement of Contributions

I hereby certify that I am the sole author of this thesis. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Part of the work described in Chapter 2 has been published in IEEE Xplore as:
U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in 2019 International Conference on Smart Applications, Communications and Networking (SmartNets), 2019, pp. 1–6.

Part of the work described in Chapters 2 and 3 has been published as:
U. Sabeel, S. S. Heydari, K. El-Khatib, and K. Elgazzar, "Unknown, atypical and polymorphic network intrusion detection: A systematic survey," IEEE Transactions on Network and Service Management, pp. 1–1, 2023

Part of the work described in Chapters 1 and 4 has been published as:
U. Sabeel, S. S. Heydari, K. Elgazzar, and K. El-Khatib, "Building an intrusion detection system to detect atypical cyberattack flows," IEEE Access, vol. 9, pp. 94352–94370, 2021.

Part of the work described in Chapters 1, 5, 6, 7, and 8 has been published

in IEEE Xplore (in the proceedings of IEEE Global Communications Conference (GLOBECOM)) and is under review in IEEE Transactions on Machine Learning in Communications and Networking:

U. Sabeel, S. S. Heydari, K. Elgazzar, and K. El-Khatib, "Cvae-an: Atypical attack flow detection using incremental adversarial learning," in 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6.

U. Sabeel, S. S. Heydari, K. El-Khatib, and K. Elgazzar, "Incremental adversarial learning for polymorphic attack detection," manuscript submitted to IEEE Transactions on Machine Learning in Communications and Networking, 2023.

Part of the work described in Chapters 1, 5 and 7 has been published in IEEE Xplore as:

U. Sabeel, S. S. Heydari, K. Elgazzar, and K. El-Khatib, "Analyzing the quality of synthetic adversarial cyberattacks," in 19th International Conference on Network and Service Management, 2023, pp. 294–298

'To my beloved daughter, Maira, whose joyous presence fueled my courage in completing this endeavor. Deep-felt thanks to my husband, Wajahat, for his unwavering patience and support. Special gratitude to my mom for her heartfelt prayers in helping me achieve my goals.'

Acknowledgements

I want to express my deepest appreciation to my research supervisor, Dr. Shahram Shah Heydari, and co-supervisor, Dr. Khalil El-Khatib, for their invaluable guidance, and continuous support throughout my research.

Dr. Heydari has been an exemplary mentor, offering not only academic insights but also encouraging a positive collaborative research environment. I am truly thankful for our weekly research discussions, his commitment to excellence, and his feedback on my academic and professional development.

I would also like to thank Dr. El-Khatib for his insightful contribution to the betterment of my research and future career prospects. His expert guidance has been instrumental in ensuring the quality of this research.

My acknowledgment also extends to Dr. Khalid Elgazzar, whose classes sparked my interest in machine/deep learning. His continuous support through my research and his constructive criticism have played a crucial role in the success of my research.

I appreciate the ongoing support from my supervisory committee and the Faculty of Business and Information Technology throughout this research. Special thanks to NSERC, the donors of the Ontario Graduate Scholarship (OGS), and Ontario Tech University for their financial assistance during my doctoral studies. Thanks to everyone who has been a part of my academic journey, both directly and indirectly.

Finally, I am grateful for the blessings of God that enabled me to successfully complete my Ph.D. study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, the use of network communication, web-based services, technologies such as voice and video over the internet, Internet of Things (IoT), cloud services, and personal computing has massively increased. Cybersecurity analysts have predicted that by the year 2025, the amount of data generated worldwide will reach around 200 zettabytes [10]. Half of this data worldwide will be stored on cloud infrastructures [10]. The availability of this information on private or public cloud infrastructure has increased the likelihood of attackers gaining illegitimate access by finding out loopholes and stealing the data. Advanced attackers can employ AI-based attack tools such as *Shortly*, to independently adjust and refine their attack tactics [11]. These advanced attacks pose a significant threat to the current network security landscape. According to the report published by cybersecurity ventures [10], the losses inflicted by cybercriminals worldwide will increase to an annual of 10.5 trillion US dollars by the year 2025. With this growth in the digital surface and increased sophistication of cyberattacks, improving the security of networks has become the focus of research

efforts.

Network specialists employ several tools to provide security to networks and connected devices such as antivirus software, firewalls, and Intrusion Detection System (IDS) [12]. An IDS is used to monitor the network for any illegal and unauthorized access which compromises the security of the network. With immense technological advancements and extensions in network sizes, the security of networks and devices has become a challenging task for traditional intrusion detection systems. To improve the network attack detection rates, researchers have investigated the use of Artificial Intelligence (AI) techniques such as Machine Learning (ML) and Deep Learning (DL) for intrusion detection. These techniques provide powerful solutions for improving cybersecurity in general and intrusion detection systems in particular due to improvement in attack detection rates as well as the reduction in false positives and negatives [13], [14]. The main advantage of these models is their ability to learn the features associated with an input and, with that prior knowledge, create a generic view of the target output for future predictions.

In the case of cyber attack identification, the pre-trained AI model inspects the input network traffic to detect any abnormal patterns by examining the changes in features such as data rate, incomplete requests, or inter-arrival times from a source under observation. A significant amount of research on intrusion detection using ML models over the past decade [15–22] has demonstrated the efficiency of such techniques in identifying known attacks with remarkable success, however, it has also been shown that the ability of ML models in detecting new, atypical and polymorphic attacks is rather limited and inefficient [1, 2, 23, 24].

One solution for this problem is to develop a practical AI-based IDS that can identify rapidly evolving atypical/ polymorphic network attacks. With the recent advances in adversarial learning based approaches especially in the field of image

processing [25–30] improvement in the results of the AI-based IDS models can be expected if similar approaches are applied to cybersecurity.

1.2 Problem Description

Most of the ML and DL-based techniques used so far for intrusion detection are based on supervised learning of their models on benchmark data with splits for training, validation, and testing phases to detect similar future attack patterns. For example, a dataset is divided into three parts- 70% for training, 20% for validation, and 10% for model evaluation. Such models are not generalized on any external test data that may contain unknown or new attacks [31–33].

Since the AI models are trained only with a specific benchmark attack dataset and network parameters, training, and testing data may share similar tool-dependent characteristics such as packet size distributions, port numbers, bandwidth, and data rates. For these systems, it is assumed that the future data has the same probability distribution as training data. These models are still limited in their abilities to secure the network against attacks because they search for specific details (like the annotated training data) that they have seen before. The main problem with such models is that they produce biased results when classifying atypical/polymorphic network attacks. This is because polymorphic attackers can use sophisticated tools to mutate attack patterns dynamically by changing multiple features such as data rates and port numbers. These sophisticated attacks are designed to use benign features that are not yet identified by the IDS, and would continuously change their features to evade detection [34].

Class imbalance in the training datasets is another main problem that impedes the performance of the current IDS [35]. It occurs when the distribution of samples of

several classes in the dataset is skewed. For example, in a network security dataset, there are only hundreds of samples for the minority attack class and millions of samples for the majority normal class. Most of the DL models are designed based on the assumption of giving equal importance to each class in the dataset. If there is an abundance of samples of one class, it becomes considerably challenging for the DL model to learn the characteristics of the minority class. It can even skew the decision of the classifier leading to overfitting. This issue makes it difficult for the DL-based IDS to analyze the attack traits effectively leading to an overall unsatisfactory attack detection performance.

Several class balancing techniques such as Random Over Sampling (ROS), Random Under Sampling (RUS), Synthetic Minority Oversampling Technique (SMOTE) [36], and Adaptive Synthetic Sampling (ADASYN) [37] have been adopted by cybersecurity researchers to resolve this problem. Abdulhammed *et al.* [38] compare undersampling and oversampling-based approaches for IDS to handle the class imbalance issue. However, RUS can result in the deficiency of conceivably important information about the input benign observations. ROS raises the quantity of minority/undersampled attack observations which can cause overfitting. Other researchers have reported an improved detection performance of AI-based IDS against undersampled attacks using SMOTE [39–41] as well as ADASYN [42, 43]. However, SMOTE depends upon interpolation to conduct oversampling which may cause the new attack samples to be less representative [44]. ADASYN, on the other hand, can synthesize minority attack instances that look like benign instances which results in high false alarms.

The evaluation metrics employed should be able to give an effective measure of model performance by classifying both normal and anomalous behavior [45]. The standard evaluation metrics used for intrusion detection systems include Accuracy,

Precision, Recall, F1_score, and Receiver Operating Characteristic (ROC) curve. Employing only a single metric such as accuracy to measure the IDS performance is not ideal and can give biased results especially when the classes in the dataset are imbalanced [46]. For example, if our binary dataset consists of 99% normal instances and only 1% network anomalies, and the IDS cannot identify any anomalies, the overall classification accuracy of the system is 99% even though no network anomalies are recognized. While recall measures the ability of a classifier to detect attacks, precision measures the trustworthiness of attack classification [47]. True Positive Rate (Recall), Precision, and F1_score can effectively measure the IDS performance on attack traffic but do not completely reflect its performance on normal traffic [48]. This can add bias to the overall classification ability of the system against typical, as well as unknown, atypical, and polymorphic network attacks [4]. Therefore, finding an appropriate set of evaluation metrics for measuring IDS performance is crucial.

Some prior work has been done in unsupervised anomaly-based IDS using DL techniques [49–55]. Such methods do not require annotated datasets for training. The models are trained only using normal network traffic and any other traffic is identified as an anomaly. Researchers have applied this technique for unknown network attack identification. The problem with this method is that it still has many unresolved issues such as high false positives [5]. Current researchers in other research areas such as text classification [56] have reported an increase in the classification accuracy of their models when using semi-supervised learning based approaches.

The network security research community has shown considerable interest in synthesizing adversarial attacks against AI-based IDS [7, 44, 57–68]. Multiple defense strategies are employed to enhance the robustness of a model. Adversarial training, for instance, is employed to improve the performance of AI-based IDS against adversarial attacks. However, in the cybersecurity domain, the generated adversarial

attacks must resemble realistic network attack traffic. The *quality* of adversarial attacks is determined by their ability to emulate the pattern of values of features for original data while preserving the functional network attack traits. If adversarial perturbations are added to alter the feature values without maintaining the network constraints, the adversarial attack becomes insignificant from a cybersecurity domain viewpoint. For instance, changing the value of *packet mean interarrival time* to generate a new attack without changing the *packet transmission rate* is not feasible, as the two features are correlated. Similarly, changing the features representing the statistical characteristics of network traffic is not always possible without considering their correlations.

Attack functionality is also another important factor. For example, a Denial-of-Service (DoS) attack must include a sufficiently high volume to cause resource exhaustion at the target. An AI-generated attack that does not take this fact into account, may generate an attack that is insufficiently impactful to cause any substantial consequence. The synthesized adversarial attack must be significant enough to evade a trained IDS, yet statistically close to the original attack to maintain attack feasibility and functionality. Training a DL model with impractical adversarial data can compromise the model since it learns invalid characteristics and can eventually degrade its robustness and generalization capability for real network scenarios [69].

To address the challenges in the current research, we employ a new and highly effective IDS training framework. This framework mimics an adversarial environment where an AI-based attacker is competing against an AI-based IDS using our hybrid model, Conditional Variational Autoencoder Adversarial Network (CVAE-AN) [3]. It is crucial to note that our primary objective is not to develop a toolbox for the attackers. Instead, our focus is on training an IDS to effectively identify any atypical/polymorphic attacks. Furthermore, this work delves into the investigation

and exploration of several criteria for ensuring the quality of synthesized adversarial attacks. While our primary research emphasis revolves around network intrusion detection, our scope can encompass other related areas as well.

1.3 Objectives and Contribution

In this thesis, we present an improved DL-based IDS for identifying dynamically changing attacks such as atypical/polymorphic network attacks. In this context, the term *feature profile* for an attack is defined as the range of values of features that constitute an attack [1]. For example, the feature profile for a TCP SYN DoS attack may include certain port numbers, a certain number of half-open connections, a certain number of set SYN flags, and a specific data rate. Attacks of different feature profiles can be generated by tweaking the feature range values randomly to perform a successful attack.

1.3.1 Atypical and polymorphic attacks

We introduce the concepts of typical and atypical network attacks in [2, 3, 70]. A *typical attack* is a predefined network attack known to the IDS from previous training iterations. Consequently, an *atypical attack* is a network attack with a different feature profile as compared to a typical/known attack. We define a *polymorphic network attack* [70, 71] as an atypical attack that mutates its characteristics or feature profile continuously to generate different variants of the same attack to bypass a network's detection systems while maintaining the functional nature of the attack.

Figure 1.1 represents an atypical/ polymorphic network attack scenario. As seen in the figure, the attacker launches an attack on the target network using sophisticated attack tools. The target network is equipped with a Network Intrusion Detection

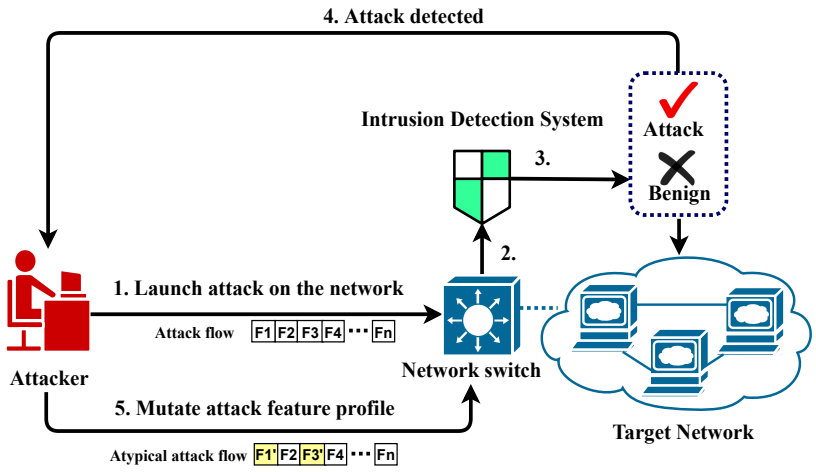


Figure 1.1: An atypical/polymorphic attack scenario. An attack with a mutated feature profile than a previously known attack represents an atypical attack. When an attacker changes the feature profile of an atypical attack continuously to evade detection by the IDS, such an attack represents a polymorphic network attack [2, 3].

System (NIDS) to identify the attack. If the attack is identified, the attacker is blocked by the target network. To validate the attack, the attacker may send a ping request to the target and observe its response. Since the attacker is blocked by the target network, this attacker mutates the feature profile to create a new atypical attack for the polymorphic attack chain. For example, $F1$ is changed to $F1'$ and $F3$ is changed to $F3'$ as shown in the figure. The goal of this attacker is to attack the network in such a way that it can evade detection from the NIDS by launching attack flows as close to benign data flows as possible. The attacker will keep on launching atypical attacks until it is successful in evading detection by the target network or until it exhausts all the input features.

Detection of such atypical and polymorphic attacks is a challenging task for an IDS. These network attacks keep on changing their patterns to enhance novelty and are hence difficult to catch by the IDS. Therefore, an improved security system to combat such attacks is the need of the hour.

We provide a thorough assessment and analysis of the proposed technique over

multiple network attack classes. We also include elaborate results using multiple applied metrics. Further, to assess the capability of the attacker, a new applied metric is employed. The IDS is examined against each class of polymorphic attacks one by one. When the IDS detects an attack, the attacker can not only change the feature profile but can also switch to a different class of attack to evade detection.

The main contributions of this research include:

- Evaluate the efficiency of existing supervised ML/DL models in identifying atypical/polymorphic cyberattacks.
- Introduce a novel CVAE-AN hybrid model designed for training the attacker and generating adversarial polymorphic attack flows. This is achieved by continuously mutating the feature profile and evading detection by the IDS.
- Implement an incremental adversarial learning-based approach for training the IDS, aimed at reducing misclassification rates for synthesized polymorphic attacks. This strengthens the performance of the IDS against such dynamically evolving atypical/polymorphic attack flows, especially those related to undersampled attacks in the training data.
- Effectively examine the performance of the IDS through the use of various applied performance metrics, including *IDS Proficiency*, *Balanced Accuracy*, and *Overall Error Rate*. These metrics provide a thorough evaluation by considering both attack and benign classifications, providing more detailed insights in contrast to other generic metrics such as F1-Score and Precision. Moreover, evaluating the attacker's effectiveness through *Evasion Success Rate* metric, which measures the success of an attack on the target IDS.
- Evaluate the performances of multiple AI models on polymorphic attack flows. Our experimental outcomes demonstrate that the proposed IDS outperforms other present-day AI models in addition to different class balancing strategies.

- Present a methodology to analyze the quality of the synthesized adversarial polymorphic attacks by comparing them with real attack data. This quality assessment ensures that attack data synthesized by our system closely resembles realistic network attacks. Additionally, it provides a comparative analysis of the quality of adversarial attacks synthesized by several state-of-the-art generative DL models.

1.4 Research Impact

Several traditional Machine Learning (ML) techniques are applied for detecting network attacks. However, increasing the accuracy of such systems against sophisticated attacks such as atypical and polymorphic attacks still needs further investigation. Some researchers [72, 73] have reported that the attack detection rate of traditional supervised AI-based IDS can be easily hampered by adversaries. Such attackers can successfully mutate the attack features to deceive the IDS which misclassifies attack data as normal. Presently, Deep Learning (DL) is gaining increasing popularity for network intrusion detection since it can handle large-scale data and identify the complex relationships among several attributes in the input data.

This work proposes an adaptive generative algorithm that can synthesize dynamic network attacks to build and train an intelligent DL-based IDS against typical and atypical/polymorphic attacks. The conditional aspect of the model provides control over the generated samples, allowing precise customization based on specific attack classes. Our technique shows significant improvements in IDS proficiency and a reduction in evasion success rates compared to other network anomaly detection techniques for multiple attack classes. The introduced attack quality metrics provide a comprehensive evaluation of the realism and effectiveness of the synthesized adversar-

ial attacks, ensuring high quality in the generated samples. Based on the improved IDS results, we expect that our algorithm can aid other cybersecurity researchers in improving the detection capability of present IDS against atypical/polymorphic network attacks. Beyond network security, the framework, coupled with generated data quality analysis, finds applications in non-security domains as well.

To the best of our knowledge, the proposed IDS is the first one of its kind, especially in the field of cybersecurity where attackers launch sophisticated attacks most of the time to surpass the current intrusion detection systems.

The subsequent sections of this thesis are structured as follows. Chapter 2 offers a foundational overview covering Intrusion Detection Systems (IDS), encompassing their taxonomy, feature engineering, and the optimization of hyperparameters to enhance AI-based IDS. Additionally, it delves into adversarial learning and explores various Deep Generative AI Models utilized in this study. Chapter 3 presents an extensive review of existing literature in the cybersecurity domain. In Chapter 4, the problem setup and underlying assumptions are thoroughly examined. Chapter 5 outlines a comprehensive framework utilizing supervised AI-based IDS against atypical/polymorphic attacks, providing insights into experimental details and evaluation outcomes. Chapter 6 describes the primary methodology developed to address polymorphic network attacks in the context of this research. The experimental design is discussed in Chapter 7, while Chapter 8 provides a detailed analysis of results for our model. Finally, Chapter 9 encapsulates the conclusion, advantages, limitations, and outlines directions for future research.

Chapter 2

Background

In the past couple of years, there have been numerous breakthroughs in intrusion detection systems (IDS). As the amount of information continues to expand exponentially and so do network attacks, advanced intrusion detection systems with ML and DL capabilities have become crucial to detect several network attacks. In this chapter, we discuss the relevant background knowledge for current IDS. We start with a typical background for IDS and discuss its taxonomy. We further discuss the importance of feature engineering and types of feature selection and hyperparameter optimization techniques employed for AI research in intrusion detection. In the end, we discuss the background for several DL models employed in this research.

2.1 IDS Background and taxonomy

An Intrusion Detection System (IDS) is a security tool that helps to analyze and identify unauthorized network or host access and any changes such as counterfeiting or destruction of data in network information systems [74]. If any abnormal activity is detected, the IDS triggers an alert to the network or host administrators.

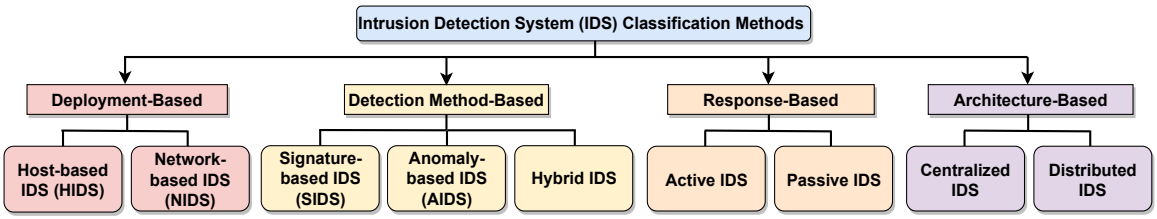


Figure 2.1: Classification of Intrusion Detection Systems based on several criteria.

An IDS can be classified based on several criteria such as deployment, detection methodology, response, and architecture. Figure 2.1 depicts the classification taxonomy of an IDS based on the above-mentioned criteria.

2.1.1 Classification based on deployment

An Intrusion Detection System (IDS) is classified into *Host-based IDS (HIDS)* and *Network-based IDS (NIDS)* based on where it is deployed.

A Network-based Intrusion Detection System (NIDS) is installed on the internal network to protect from unauthorized access and detect any attacks on the network hosts. A NIDS can be installed on a centralized switch or subnet to analyze the network traffic for any suspicious activity such as Denial of Service (DoS) Attacks. A Host-based Intrusion Detection System (HIDS), on the other hand, is installed on the host machine to monitor the traffic originating and coming into the specific host. HIDS cannot identify suspicious behavior on other hosts in the network. It is highly effective in identifying any insider threats to the host such as unusual client-server communication through certain ports, file manipulation, overwriting, or deletion. It is more challenging to install a HIDS on multiple devices since it requires extra computational overhead. For enhanced security of a network, both NIDS and HIDS need to be deployed. The NIDS can detect large-scale attacks with a very fast response time while the HIDS is capable of identifying attacks that originate from within the

network.

2.1.2 Classification based on detection method

An IDS is further classified into three major categories based on the detection technique: *Signature based IDS (SIDS)*, *Anomaly-based IDS (AIDS)*, and *Hybrid IDS*.

The SIDS is also known as *knowledge-based IDS* or *misuse-based IDS*. It compares the intrusions or attacks to the known attack patterns in its database [75]. This type of intrusion detection technique has a high true positive rate for known attack patterns due to the availability of such patterns in its database. However, it is difficult to identify unknown/atypical attacks or polymorphic attacks using this technique due to the unavailability of such attack patterns in the database. Additionally, for SIDS, it is a challenging and time-consuming task to update attack patterns for every new attack [76].

AIDS is also known as *behavior-based IDS*. AIDS is used to identify abnormal behavior by comparing it with baseline behavior and then taking action accordingly. Although this type of intrusion detection technique can detect unknown or new attacks, it has a high false positive rate when classifying normal and anomalous patterns [75]. Hybrid IDS combines the best features of both SIDS and AIDS. The main aim of employing a hybrid IDS is to reduce the misclassifications of unknown and polymorphic attacks, extract useful patterns from these attacks, and improve the true positive rates for these attacks [70].

2.1.3 Classification based on response

An IDS can be classified into *active* and *passive IDS* based on its response mechanism when an attack is detected.

An active IDS also known as an *Intrusion Prevention System (IPS)* is configured in such a way that as soon as an attack is detected, the system automatically blocks these attacks without even consulting the security analyst. Such an IDS provides a real-time response by triggering an alarm when the attack is detected, blocking the attack, generating a report, creating a backup, and logging all the information [77]. Sometimes an IPS may be susceptible to attacks such as Denial of Service (DoS). For example, if false positives and normal traffic have not properly been identified, it may lead to denial of essential services to legitimate clients because of high false positives. A passive IDS on the other hand is configured to scan and analyze network traffic and alert the network analyst to take further action such as blocking IP addresses, termination of the connection/process, and locking the user account [77]. A passive IDS is easier to configure and install and is less susceptible to attacks as compared to an active IDS [77].

2.1.4 Classification based on architecture

Based on the infrastructure requirements, an IDS can be classified as *centralized* and *distributed* [77].

In a centralized IDS, as the name suggests, IDS is installed on a central device that is responsible for analyzing network traffic and generating an alarm if any abnormal patterns are detected. This information is sent to the central device by other devices in the network. The biggest disadvantage of this system is that if the central device is hacked or is non-functioning, the entire network is susceptible to further attacks. Additionally, with the increase in network logs, the central device may get overwhelmed due to excessive overhead [77]. This centralized IDS makes independent decisions about intrusions in the network, hence may also be known as an *independent IDS*.

On the contrary, in the case of a distributed IDS, each device in the network can detect and respond to intrusions. Such an IDS follows a hierarchical tree-like architecture where each node communicates with other nodes in a bottom-up approach [77]. The distributed IDS makes collaborative decisions regarding a detected attack in the network, hence is also known as *collaborative IDS*. Some challenges faced by a distributed IDS are balancing the load, fault tolerance, and detection of insider threats [77].

In this thesis, we mainly focus on the current literature for DL algorithms employed in network intrusion detection. This is discussed in more detail in chapter 3.

2.2 Feature Engineering

Feature engineering is a process of transforming data into a format that is easily interpretable by the AI model. The data quality and features are directly proportional to the detection ability of such models. In AI research, the feature engineering phase is still an overlooked subject and needs careful examination and research [78]. Training data with a relevant feature subset requires a less complex AI model which provides better classification results.

Unsupervised AI models, on the other hand, do not require the feature selection process. They are based on the concept of dimensionality reduction which reduces the features to dimensions that explain most of the input data. The complete feature engineering workflow is depicted in Figure. 2.2. For unsupervised AI, the feature selection phase is omitted after dimension reduction.

Raw input data goes through various phases of feature engineering before it finally becomes available for AI training. The data is cleaned by removing 'nans' and

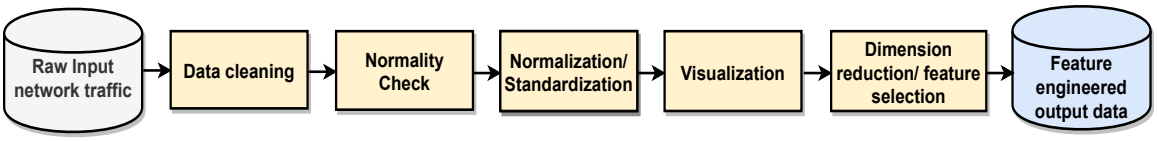


Figure 2.2: Feature Engineering Workflow

infinite values, converting negative values to null values, and conversion of categorical features like port numbers into numerical categories (say 0,1,2 and so on). Clean data distribution is inspected for normality for example using Quantile-Quantile plots (QQ plots) [79].

This plot compares the plot generated for the given dataset with the ideal Gaussian (Normal) distribution. If the data follows this bell-shaped curve, the points from both the generated and standard plots should overlap on the standardized line. Normalization of data leads to faster convergence of an AI model and avoids conditions where gradients take a long time and keep on switching back and forth before reaching global or local minimum. If the input clean data does not follow a normal distribution, it may be normalized using for example Z-Score normalization technique which changes the data distribution such that it has a mean of zero and a standard deviation of one. Normalized data is then subjected to dimensionality reduction using Principal Component Analysis (PCA) [80] or other unsupervised techniques such as Autoencoders. These techniques are useful for visualization of the high dimensional data into a lower-dimensional space for a better understanding of data distribution. The final step is selecting the best feature subset based on various techniques discussed in the following sections.

Figure. 2.3 represents the QQ plot for clean CICIDS2017 before normalization [1]. The input data distribution is inspected for normality using QQ-plots which compares different data points from given data to Gaussian distribution. The graph indicates that theoretical quantiles tend to have lower values than sample quantiles. The data

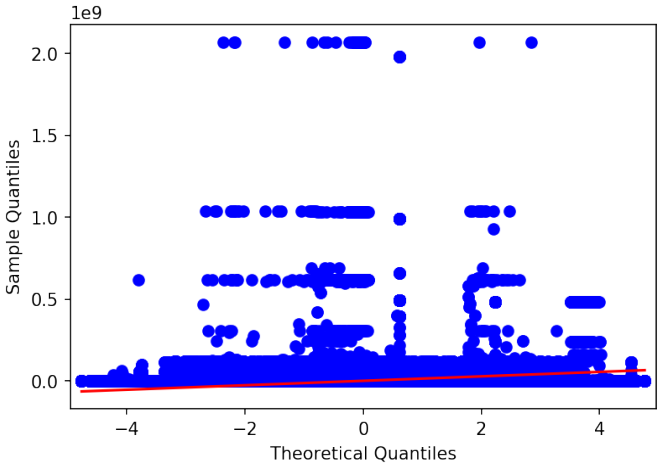


Figure 2.3: QQ Plot for Clean Data Before Normalization for CICIDS2017

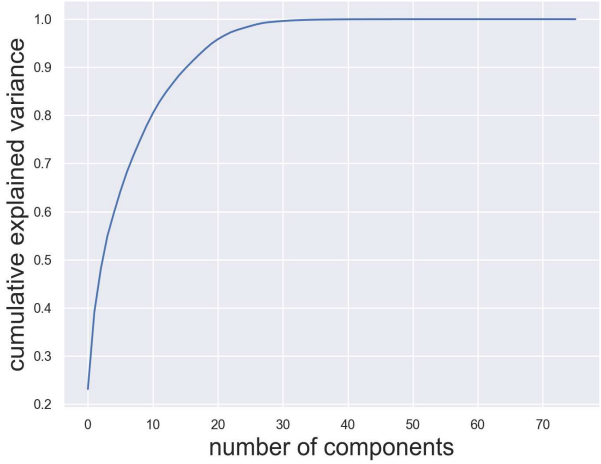


Figure 2.4: Cumulative Explained Variance Ratio for CICIDS2017 using PCA

does not follow Gaussian distribution because most of the data points or quantiles deviate from the standardized reference line. Therefore, it needs to be normalized before feeding into the AI model.

After normalization, Principal Component Analysis (PCA) can be applied to the data for visualization and a better understanding of data distribution [1]. PCA reduces the larger dataset to a smaller one following the Gaussian distribution. Figure. 2.4, represents the cumulative explained variance ratio using PCA which measures

the number of important components in the dataset. In our case [1], the first 25 principal components explain 98.20% of the total variance available in the data. In other words, only the first 25 components for input CICIDS2017 data are the most significant for AI training.

2.2.1 Feature Extraction and Dimension Reduction

Feature extraction consists of using an algorithm to choose the best-suited subset of features for a specific DL model based on intuition from the input data. It is a vital constituent of the ML/DL models employed in intrusion detection. Feature or attribute extraction allows for the reduction of biased results by removing the less relevant/noisy features that do not help in improving classification results. For example, the feature *packets per second* may be less significant in case of a low-rate Denial of Service attack such as Slowloris which mainly depends on the number of half-open connection requests rather than the amount of data sent. Feature selection also helps in reducing the training complexity and overfitting by decreasing the error on validation and test data [81]. In Figure 2.5, we provide the taxonomy of several supervised and unsupervised attribute selection techniques in intrusion detection.

Supervised feature extraction methods are further classified as *filter method*, *wrapper method*, and *embedded/intrinsic method*.

Filter Method

The filter method applies a statistical algorithm to determine the correlation of a feature with a specific target variable. Several examples of such techniques are Pearson's correlation [82], Analysis of Variance (ANOVA) [83], Chi-Square [84], and Linear Discriminant Analysis (LDA) [85]. Pearson's correlation measures the dependence between a continuous feature and a continuous target variable. ANOVA is a statistical

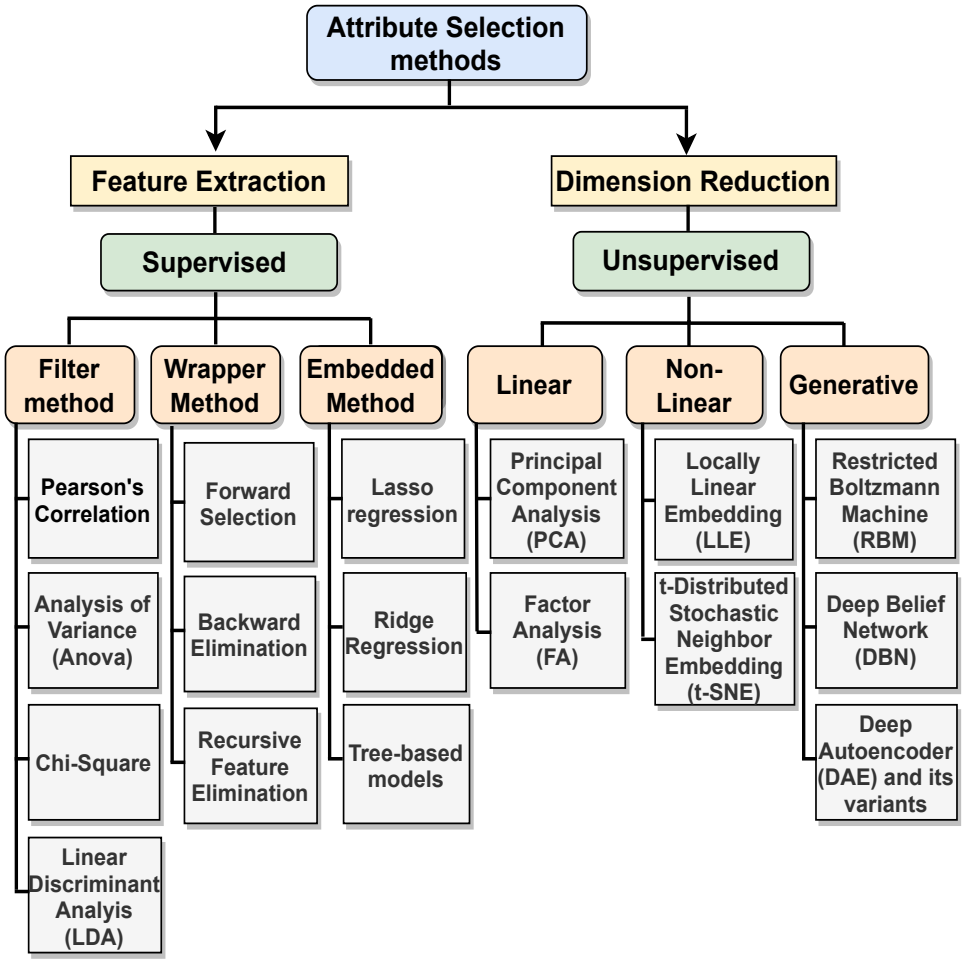


Figure 2.5: A classification of supervised and unsupervised attribute selection methods in intrusion detection

test that measures the correlation between a categorical feature and a continuous target variable. The Chi-Square method measures the dependence between a categorical feature and a categorical target variable. LDA finds the linear combination of continuous features to classify two or more classes or target variables. In [86], the authors apply an ensemble of filter feature selection techniques such as Chi-Square, Pearson's correlation, and Mutual Information to reduce the complexity of training an IDS and maintaining the performance of the system. Other intrusion detection research works employ LDA [87] and ANOVA [88] for feature selection to decrease

the complexity of the IDS model and achieve faster responses.

Wrapper Method

The wrapper method selects the most suitable features that improve the classification results of a machine learning algorithm. Forward Selection (FS) is a wrapper method that commences with a set of null features and keeps on adding the best-suited feature after every iteration. Backward Elimination (BE), on the other hand, starts with a full set of features and after every iteration removes the worst-suited feature from the remaining set of features. Recursive feature elimination (RFE) is another wrapper method that employs the greedy algorithm to identify the best-suited feature subset. At each iteration, the best or worst-suited features are identified which are then organized based on the order of elimination. In [89], the researchers propose a Meta-Heuristic-based Sequential Forward Selection (MH_SFS) feature selection algorithm to reduce the high dimensionality problem in the training dataset and improve the results of anomaly detection models. Al-Jarrah *et al.* [90] apply both FS and BE to select features for training an IDS model and effectively improve the detection rate using the KDD '99 dataset. The authors in [91] employ RFE to identify the most effective features for attack classification using their Deep Neural Network (DNN).

Embedded Method

The embedded method combines the best characteristics of both filter and wrapper methods. These are applied in certain machine learning algorithms that have built-in or intrinsic feature selection. In this case, during each training iteration, the best-suited features are extracted. Examples of embedded feature selection methods include certain regularization algorithms that shrink the data values such as Lasso Regression [92], Ridge Regression [93], and tree-based algorithms such as Decision

Tree (DT) and Random Forest (RF). In [94], the authors apply undersampling and embedded feature selection based on Light Gradient Boosting Machine (LightGBM) for multi-classification of attack classes and normal traffic in the network. Linear models such as Logistic Regression (LR) and Linear Support Vector Classifier (L-SVC) select features by using regularization for overfitting. This causes features that have a weak association with the target class to shrink to zero and therefore eliminated. The equation for regularization in LR is represented in eq.(2.1).

$$S(w) = argmin \sum_{i=1}^{n} log(1 + e^{a}) + \lambda ||w|| \tag{2.1}$$

Here the value of $S(w)$ should be minimized (almost zero) and therefore, w will be equal to infinity. Therefore, $L1$ regularization is added which is represented by the second right term in eq.(2.1). If λ increases, regularization increases which in turn decreases overfitting in LR. w represents the vector perpendicular to the hyperplane separating attack and normal classes. If x_i is the training data and y_i is the target label, then a is represented by $-y_i w^T x_i$.

The regularization equation in L-SVC is represented in eq.(2.2). Here $||.||_1$ represents 1-norm and C is the regularization parameter whose value should always be greater than zero. If C value is smaller, the number of features selected is less, and vice versa.

$$S(w) = min||w||_1 + C \sum_{i=1}^{n} (max(0, 1 - y_i w^T x_i))^2 \tag{2.2}$$

Tree-based models such as Random Forest Classifier (RFC) and Extra Tree Classifier (ETC) select features based on the mean decrease in impurity known as Gini index [95]. The features with the least impurity are used for splitting the nodes. The importance score of a feature x_i over all the m trees in random forest is given in

eq.(2.3). Here $S_m(X_i)$ is the importance score of the feature x_i for one Decision Tree T_m.

$$S_T(X_i) = 1/m \sum_{m=1}^{m} S_m(X_i) \tag{2.3}$$

Ensemble Gradient Boosting models such as the Light Gradient Boosting Model (L-GBM) find the most relevant feature based on some threshold from a local set of features in each tree. The selected feature with the highest score indicates its usefulness in the construction of the tree. This process is repeated until we get the final feature subset [96]. The final loss minimization equation for L-GBM is given in eq.(2.4).

$$L_{min} = min_{t\epsilon T} \sum_{i=1}^{n} h_i/2 \left(t(x_i) + g_i/h_i\right)^2 \tag{2.4}$$

Here $t(x)$ is a tree in the ensemble, x_i are the training samples, $t(x_i) = y_i$ is the target class, h_i is the hessian of the loss on sample i, g_i is the gradient of the loss on sample i.

Table 2.1 provides a comparative analysis of current supervised feature extraction techniques in intrusion detection based on several criteria.

Dimension reduction employs an unsupervised learning technique to reduce the number of features while maintaining as much variance in the input data as possible. Working with high-dimensional data is not always desirable for many reasons such as computational complexity and slow convergence time. Therefore, dimension reduction is applied by researchers to reduce the number of attributes in the input data in multiple research areas such as image processing [97], speech recognition [97], bioinformatics [97], and cybersecurity [98–100]. In this work, we identify several popular unsupervised dimension reduction techniques for intrusion detection into 3 classes: *Linear*, *Non-Linear*, and *Generative*. Some examples are Principal Component Anal-

Table 2.1: Comparison of Supervised Feature Extraction Techniques in Intrusion Detection

Criterion	**Filter Method**	**Wrapper Method**	**Embedded Method**
Main idea	Applies a statistical algorithm to determine the correlation of a feature with a specific target variable.	Selects the most suitable features which improve the classification results of machine learning algorithm under evaluation.	Applied during training in certain machine learning algorithms that have built-in or intrinsic feature selection.
Computational time	Faster	Slower	Medium
Scalability to high dimensions	Yes	No	Yes
Overfitting	No	High	Low
Generalization	Good	Better than Filter	Better than Filter
Classifier Dependence	No	Yes	Yes
Modeling Feature dependencies	No (in case of univariate)	Yes	Yes
Feature selection performance	Best features are not always selected	Better	Good
Examples	Pearson's correlation ANOVA Chi-Square LDA	Forward Selection Backward Elimination Recursive Feature Elimination	Lasso Regression Ridge Regression Tree-based models
References	Karna *et al.* [86]: Chi-Square, Pearson's correlation, and Mutual Information Attia *et al.* [87]: LDA Shakeela *et al.* [88]: ANOVA	Liu *et al.* [89]: Forward Selection Al-Jarrah *et al.* [90]: Backward Elimination Ustebay *et al.* [91]: Recursive Feature Elimination	Hua *et al.* [94]: LightGBM

ysis (PCA), Factor Analysis (FA), Locally Linear Embedding (LLE), t-Distributed Stochastic Neighbor Embedding (t-SNE), Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN), Deep Autoencoder (DAE) and its variants.

Linear dimension reduction

PCA is a linear unsupervised dimension reduction technique in which features from a high-dimensional space can be projected to a lower-dimensional space while preserving as much variance as possible [80]. This makes it easier to explore and analyze a dataset with lower dimensions and reduces the overall computational complexity of the DL model. FA is a linear unsupervised dimension reduction statistical technique employed to reduce the number of features in the original dataset into a reduced feature set known as factors. Each factor has an influence on the variance, but some

factors influence variance more than others [101]. Several cybersecurity researchers employ PCA [1, 102, 103] and FA [104] for dimension reduction and improving the classification results for their AI-based IDS.

Non-linear dimension reduction

LLE is a non-linear unsupervised dimension reduction technique that converts high-dimensional data to a low-dimensional representation by preserving the local distances among data points. The local projections are then globally compared to identify the finest non-linear embedding [105]. Researchers in [106, 107] apply LLE to reduce the dimensions of input data before training the AI algorithm for intrusion detection. Their results confirm improvement in detection rates and reduction of false positive rates. t-SNE is a non-linear unsupervised dimension reduction technique that is used for visualizing high-dimensional data into a lower-dimensional space. Similar data points are assigned joint probabilities on a higher-dimensional space. t-SNE assigns a similar probability distribution over the data points in a lower-dimensional space and tries to minimize the distance between the two probability distributions [108]. In [109, 110], the authors apply t-SNE dimension reduction to improve the attack classification accuracy using an AI-based IDS.

Generative dimension reduction

RBM is a generative unsupervised neural network that reduces the dimensions of the input feature vector by learning the probability distribution of the input data and retaining the most relevant information [111]. A DBN [112] is created by stacking multiple RBMs to create a deeper neural network. It is applied for unsupervised dimension reduction and works similarly to RBM. DAE is a generative unsupervised deep neural network that learns the most important variables in the dataset by condensing

Table 2.2: Comparison of Unsupervised Dimension Reduction Techniques in Intrusion Detection

Criterion	**Linear Method**	**Non Linear Method**	**Generative Method**
Main idea	Generate a linear mapping of data in a lower dimensional plane from a higher dimensional plane while preserving important information in the data.	Generate a non-linear manifold by preserving local distances between training instances in the original space.	Generate a latent representation from a higher dimensional space by employing joint probability distribution.
Computational time	Low	High	High
Complexity	Low	High	High
Noise sensitivity	Low	High	High
Large training dataset	No	No	Yes
Linear relationships between variables	Yes	No	No
Preserves local relationships	No	Yes	No
Overfitting	No	Yes	Yes
Hyperparameter optimization	No	Yes	Yes
Examples	PCA FA	LLE t-SNE	RBM DBN DAE and its variants
References	Sabeel *et al.* [1]: PCA Meng *et al.* [102]: PCA Zhao *et al.* [103]: PCA Wu *et al.* [104]: FA	Zheng *et al.* [106]: LLE Hou *et al.* [107]: LLE Hamid *et al.* [109]: t-SNE Yao *et al.* [110]: t-SNE	Seo *et al.* [100]: RBM Alom *et al.* [114]: RBM Elsaeidy *et al.* [115]: RBM Alrawashdeh *et al.* [98]: DBN Zhao *et al.* [116]: DBN Li *et al.* [99]: DAE Qureshi *et al.* [49]: DAE

the information in the input data into a low-dimensional space or latent space in the hidden layer [113]. Several cybersecurity researchers employ RBM [100, 114, 115], DBN [98, 116], and DAE [49, 99] for dimension reduction of an input dataset combined with other classifiers for intrusion detection. Table 2.2 provides a comparative analysis of unsupervised dimension reduction techniques in intrusion detection based on several criteria.

2.3 Hyperparameter Optimization (HPO)

A hyperparameter (HP) unlike a regular model parameter for a DL model is a criterion that cannot be learned during the training process but has to be tuned before

training. Some common examples of hyperparameters are learning rate, batch size, number of hidden layers for a DL model, number of neurons for each layer, and type of regularization. HPO refers to the process of building an efficient DL model after applying an algorithm to attain an optimal model structure by modulating its hyperparameters [117]. HPO is an important component for building a model with multiple hyperparameters such as a Deep Neural Network (DNN) [118]. Tuning the right set of hyperparameter combinations allows for minimizing the cost or error function of the model on validation and test data. Finding the most appropriate set of hyperparameters for a DL model is an intensive task since the performance of the model is sensitive to even small variations in the hyperparameters and may lead to overfitting or underfitting problems. The general equation for HPO [119] is given in eq. (2.5).

$$x^{'} = \underset{x \epsilon X}{\operatorname{argmin}} f(x) \tag{2.5}$$

Where $f(x)$ is the objective function (loss or error function) that needs to be minimized, $x^{'}$ is the selected set of hyperparameter combinations that help to minimize the error function, and x is any hyperparameter combination that belongs to a list of X hyperparameters.

The commonly employed hyperparameter optimization techniques for improving a DL model include Manual Search, Grid Search, Bayesian Optimization, and Evolutionary Optimization.

2.3.1 Manual Search

The Manual Search method is a traditional trial-and-error method for hyperparameter tuning. A set of hyperparameters is selected by the DL engineers, the model is trained with these hyperparameters, and the performance is measured in terms of the loss

function. The hyperparameters are then adjusted and the process is repeated until the loss or error of the model is minimized. This is a complex technique especially if there are a large number of hyperparameters to choose from. In cybersecurity, manual hyperparameter optimization to select the best hyperparameters for IDS models is the most commonly used technique [2, 120–124].

2.3.2 Grid Search

In the Grid Search method, a list of possible hyperparameter values is created. Separate DL models are built for every possible combination of the hyperparameters from the list provided. Each model is then evaluated and the architecture that provides the best results is selected. The Grid Search method is highly computationally expensive due to the high dimensions of the hyperparameters involved just like the traditional Manual Search technique. Some cybersecurity researchers apply Grid Search Hyperparameter optimization to select the best set of hyperparameters for training their specific IDS models [125–130].

2.3.3 Random Search

Just like Grid Search, in the Random Search method, initially, a list of possible hyperparameter values is created. During each iteration, a random combination of hyperparameters is selected from the list. A DL model is built based on this randomly selected hyperparameter combination and its performance is recorded. In the end, the hyperparameter combination that provides the best model performance is selected. Although Random Search provides better performance as compared to Grid Search, it is still computationally expensive. In Figure 2.6, we provide a comparison of the Grid Search and Random Search Optimization techniques. Several DL models

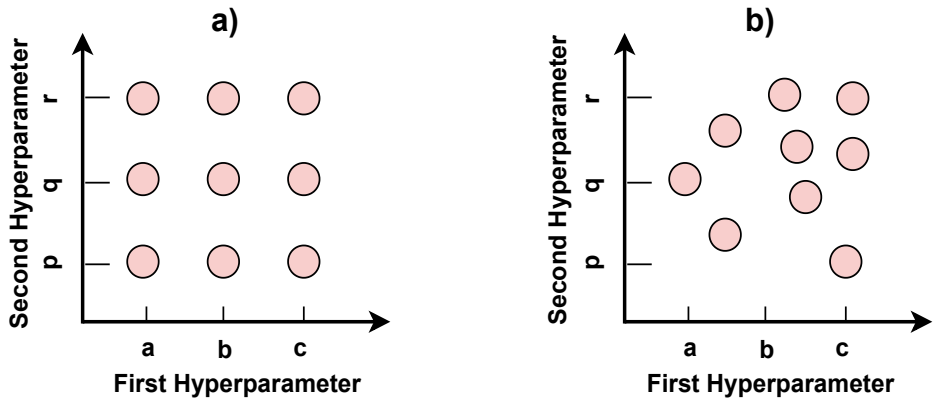


Figure 2.6: An illustration of Grid Search (a), Random Search (b). The grid search algorithm employs an exhaustive search for every possible hyperparameter combination. For Random search, the parameters to train a DL model are sampled randomly.

such as DNN [131,132], DAE-DNN [133], Convolutional Neural Network (CNN) [134], Convolutional Long Short-Term Memory (Conv-LSTM) [135,136], and Gated Recurrent Unit (GRU) [137] employed by network security researchers use Random Search to select the best hyperparameters and provide better classification results for attack detection.

2.3.4 Bayesian Optimization

Bayesian Optimization is a type of Sequential Model-Based Optimization (SMBO) algorithm [138] which allows the information of one iteration to be used in the next for improving the overall results. It reduces the number of iterations of model training and evaluation since it only employs the hyperparameters that are expected to give better evaluation results. This is unlike the computationally expensive Grid Search and Random Search techniques where at each iteration, individual models are built using a different set of hyperparameters and the result of each iteration cannot be used for the next. Bayesian optimization is based on the Gaussian process that constructs the posterior probability distribution for optimization [139]. Initially, the

model is built, trained, and evaluated with a certain set of hyperparameters. In the next iteration, the hyperparameters from the previous phase are used to compute the posterior distribution for building the model in this phase. This process is repeated iteratively to achieve the optimum. Many research works in network security [140–144] employ Bayesian Optimization to select the best hyperparameter set for their DL-based IDS models. This ensures that these models are well trained and do not overfit or underfit the evaluation data.

2.3.5 Evolutionary Optimization

Evolutionary optimization employs the concept of natural evolution to solve an optimization problem, especially in the case of dynamically changing environments [145]. It employs selection, crossover, and mutation phases for hyperparameter selection. In the first phase, an initial population of hyperparameters is randomly chosen and organized based on their objective function (fitness function) which is designed to achieve the desired outcome. The poorly performing hyperparameters from the first phase are then replaced with new hyperparameters created during the crossover and mutation phases between the hyperparameters selected in the first phase. These new hyperparameters then advance to the next generation. This process is repeated iteratively until the desired approximation is reached. Evolutionary optimization does not make assumptions about the underlying relationships between different variables in the observed data. Hence, such techniques are free of human biases and produce varying results.

Figure 2.7 depicts the comparison of Bayesian and Evolutionary algorithms for HPO. Bayesian optimization constructs a distribution of functions named Gaussian Process which describes the optimization (Acquisition) function at every step. The process is repeated iteratively and is refined by sampling more interesting regions in

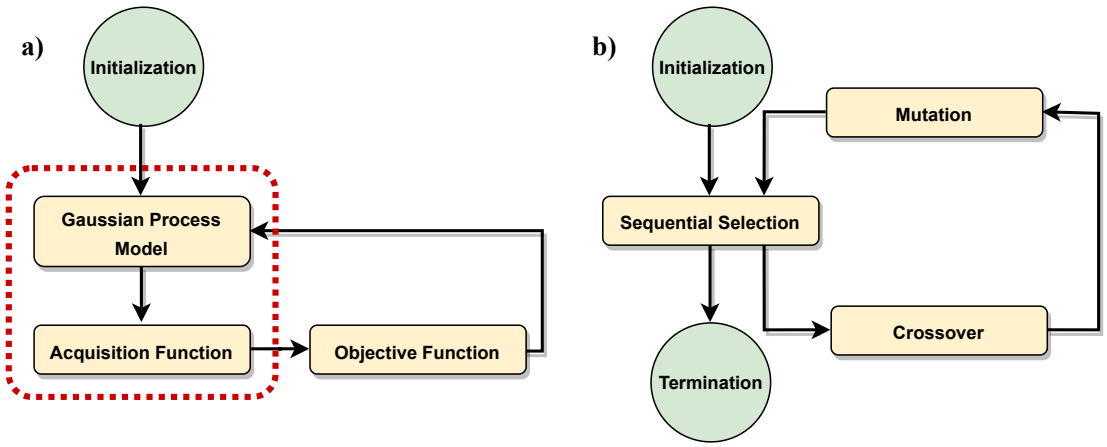


Figure 2.7: (a) Bayesian HPO (b) Evolutionary algorithm for HPO.

the hyperparameter space. Evolutionary optimization starts with an initial set of hyperparameters which are then replaced by a new set after crossover and mutation phases. This process iteratively repeats itself until the desired optimum is reached.

Schubert *et al.* [142] adopt automated machine learning techniques such as Tree-Based Pipeline Optimization Tool (TPOT) for intrusion detection. This technique employs Evolutionary hyperparameter optimization to identify the best hyperparameter set for training. Other researchers also apply Evolutionary optimization to train their IDS models such as DNN [141, 146], CNN [147], Long Short-Term Memory (LSTM) [146], CNN+BiLSTM hybrid [148], GRU [146], and DBN [146] effectively.

We have provided a comparative analysis of several state-of-the-art hyperparameter optimization techniques based on certain criteria in Table 2.3. The table also includes a comparison of multiple DL algorithms in intrusion detection for each type of hyperparameter optimization technique employed in these research works.

Table 2.3: Comparison of Several State-of-the-art Hyperparameter Optimization Techniques in Intrusion Detection

Criterion	**Manual Search**	**Grid Search**	**Random Search**	**Bayesian Optimization**	**Evolutionary Optimization**
Open-source libraries	N/A	Scikit-learn [149] Talos [150] Katib [151] H2O AutoML [152] Determined [153] Tune [154]	Scikit-learn [149] Talos [150] Katib [151] Hyperopt [155] Determined [153] Tune [154]	Scikit-optimize [138] Auto-sklearn [156] Katib [151] Optuna [157] SMAC [158] Ax [159] tuneRanger [160]	DEvol [161] Deap [162] Nevergrad [163] Determined [153] Tune [154] TPOT [164,165]
Computational Complexity	High	High	High	Low (if dimension is low)	Low (due to approximation)
Parallel training	Yes	Yes	Yes	No	Yes
Knowledge of previous iterations	No	No	No	Yes	Yes
Works well in a dynamic environment	No	No	No	No	Yes
DL Algorithm in intrusion detection	DNN [2, 120] CNN [121–123] LSTM [122, 124]	DNN [125, 126] CNN [127] LSTM [128, 129] ICVAE-DNN [130]	DNN [131, 132] DAE + DNN [133] CNN [134] Conv-LSTM [135, 136] GRU [137]	DNN [140, 141] AutoML [142, 143] AE, DAE [144]	TPOT [2, 142] DNN [141, 146] CNN [147] LSTM [146] CNN + BiLSTM [148] GRU [146] DBN [146]

2.4 Adversarial Machine/Deep Learning

Adversarial machine learning is an emerging area of research that aims to assess and enhance the resilience of ML/DL models against deceptive behaviors [166]. Adversarial attack examples are intentionally crafted inputs designed to evade detection by an AI-based IDS. As shown in Fig. 2.8, an adversarial attack example $a^{'}$ can be generated using a legitimate input a by adding a small and precise perturbation λ to it. The value of λ should be significant enough to cause the IDS to misclassify a legitimate attack as normal while maintaining the functional nature of the attack.

Most ML/DL algorithms are designed on the assumption that train and test data come from the same source and hence share the same statistical characteristics [1–3, 70]. This assumption creates a vulnerability for such systems as attackers might

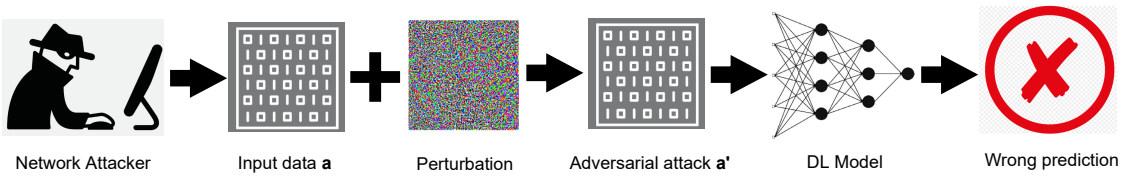


Figure 2.8: Adversarial Machine/Deep Learning in cybersecurity.

deliberately provide falsified data that deviates from the statistical distribution to gain access to the system. Adversarial attacks have been categorized into the following categories [167]:

- *Poisoning attack*: This type of attack is implemented on the training data of an ML/DL model. This attack intends to impair the model's ability to precisely identify accurate labels. For example, an attacker can change attack labels to normal labels to evade detection when the model is trained on the perturbed data.

- *Evasion attack*: This attack mainly targets the model itself by exploiting its vulnerabilities after training. By making small perturbations in the input data, the model is manipulated to make incorrect predictions. For example, an attack data sample can be perturbed to look like normal data by hiding the IP address that was previously blocked or changing the range of any other network attack feature to look like a normal data feature.

- *Model extraction attack*: This attack mainly focuses on stealing model information such as its structure, parameters, or training data. The attacker may reproduce the same model for its benefit to facilitate the execution of subsequent attacks. For example, an attacker can steal data and model parameters for a spam filtering ML/DL model. Using this information, the attacker may identify spam keywords and could manipulate a spam email to guarantee its

successful delivery to the recipient's inbox.

- *Inversion attack*: This attack centers on obtaining sensitive information about training data by analyzing the predictions of an ML/DL model. For example, an attacker can examine the predictions of an AI-based IDS to infer if a particular data flow belongs to the training data for that model [168].

The main focus of this research is on evasion attacks that are synthesized using generative DL models.

2.4.1 Insights of Generative DL Models

This section provides a general description of different generative deep learning components such as Generative Adversarial Network (GAN), Variational Autoencoder (VAE), Conditional Variational Autoencoder (CVAE), and Semi-supervised GAN (SGAN) employed for this work.

Generative Adversarial Network (GAN)

GAN is a generative deep learning method consisting of two sub-models: Generator (G) and Discriminator (D) in competition with one another [169]. The generator is trained to gain useful insights from input data patterns and later synthesize similar plausible output samples. The discriminator classifies samples from real data distribution (input data) and synthesized data (from the generator). The role of the generator is to generate data samples similar to input data to elude discovery by the discriminator. Conversely, the role of the discriminator is the binary classification of real input data from synthesized data. Training of both the discriminator and the generator continues until Nash equilibrium is achieved. Fig. 2.9 shows the architecture of a basic GAN model.

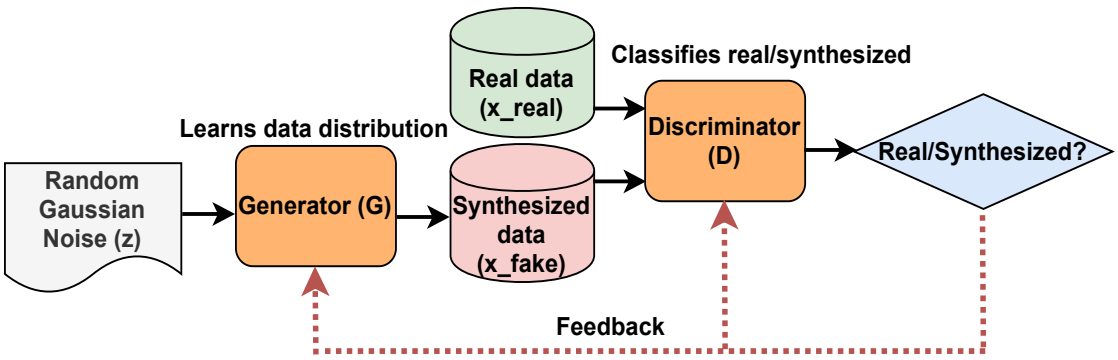


Figure 2.9: Architecture of a basic Generative Adversarial Network (GAN)

The loss function for the Discriminator in the GAN model is given in eq.(2.6).

$$L_D = \max_D [\mathbb{E}_{x \sim p(x)}[log\ (D(x))] + \mathbb{E}_{z \sim p(z)}[log(1 - D(G(z)))]] \tag{2.6}$$

Here, x represents real data samples, $p(x)$ represents real input data distribution for training the discriminator D, E_x is the expected value over all real data samples. z represents the Gaussian noise vector, $p(z)$ is the input Gaussian distribution for training the generator G which synthesizes samples $G(z)$, E_z is the expected value over all synthesized data samples. $D(x)$ represents the probability estimate of the discriminator that a real data sample is real and $D(G(z))$ represents the probability estimate of the discriminator that a synthesized data sample is real.

The training of generator G is dependent on the discriminator D making a mistake by classifying synthesized data as real [169]. Therefore, the generator tries to minimize $1 - D(G(z))$ which represents the probability estimate of the discriminator that a synthesized data sample is fake. Eq.(2.7) represents the loss function for the generator.

$$L_G = \min_G [\mathbb{E}_{z \sim p(z)}[log(1 - D(G(z)))]] \tag{2.7}$$

During the training, the Discriminator tries to learn its best whereas the Generator

tries to evade this best Discriminator. The overall loss function for a GAN model is given in eq.(2.8).

$$L_{GAN} = \min_{G} \max_{D} [\mathbb{E}_{x \sim p(x)}[log D(x)] + \mathbb{E}_{z \sim p(z)}[log(1 - D(G(z)))]] \tag{2.8}$$

The main aim of the discriminator is to increase the distance between $D(x)$ and $D(G(z))$ to the maximum for identifying real and synthesized data. On the other hand, the main aim of the generator is to decrease the distance between $D(G(z))$ and $D(x)$ to the minimum for generating plausible samples similar to input data.

Variational Autoencoder (VAE)

A traditional Deep Autoencoder (DAE) is a Deep Neural Network (DNN) model mostly used by researchers for unsupervised dimension reduction and feature selection [99]. The main objective of this model is to extract important features from input data and reconstruct the lower-dimensional data again at the output layer under some constraints. The model consists of two components namely Encoder (E) and Decoder (D). The role of the encoder is to take the input data and transform it into a lower-dimensional latent representation. The decoder on the other hand uses the latent representation from the encoder to reconstruct the input. The reconstruction loss between the actual input and the low dimensional output generated by the DAE gives the measure of how far or different the generated output data is from the actual input data. Figure. 2.10 shows the basic architecture of a DAE.

The encoder function is represented as $z = f(x)$ and the decoder function as $x' = g(z)$. Here, x represents the input, z represents the internal latent representation and x' represents the output. The role of the encoder is to map the input x to the output x' through the latent code z.

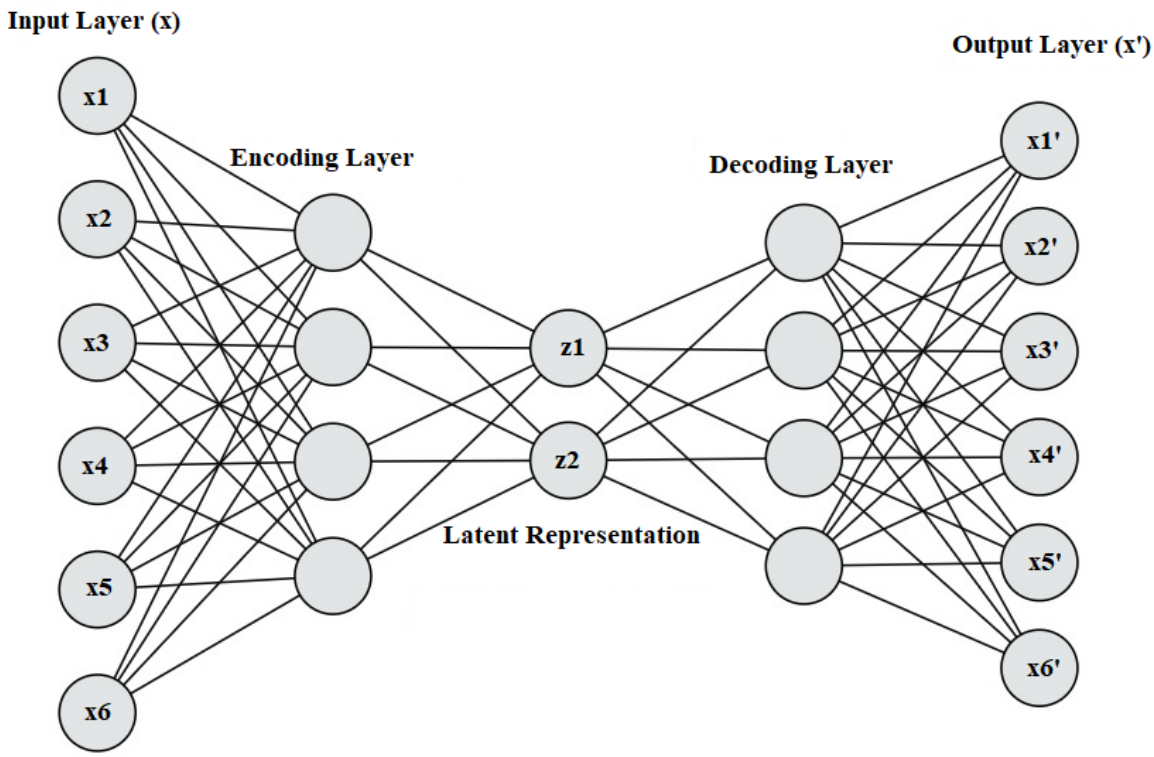


Figure 2.10: Basic Architecture of a DAE

Since a simple DAE does not have generative properties, it cannot generate data that it has never seen before. To solve this problem, the VAE was introduced by Kingma *et al.* [170]. Unlike a vanilla DAE which generates the latent representation from the previous hidden layer, a VAE uses the mean and standard deviation from all the previous hidden layers to create the latent distribution like a Gaussian distribution. The decoder helps to synthesize the input data from the latent variable. This model learns to synthesize new data as well as learn from the latent vector. Training is done using the Stochastic Gradient Variational Bayes estimator [170] which helps the encoder to learn the approximation of posterior distribution. Figure. 2.11 represents the basic architecture of the VAE model.

The main objective, in this case, is to minimize both the reconstruction loss as well as latent loss. The loss function for a VAE is represented in eq.(2.9) where θ and

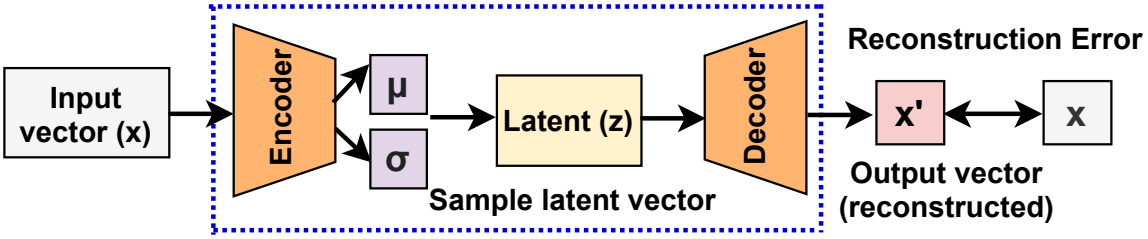


Figure 2.11: Architecture of VAE

ϕ represent the parameters for the encoder and decoder, respectively.

$$L(\phi, \theta) = \mathbb{E}\left[log\, p_\theta(x|z)\right] - D_{KL}[q_\phi(z|x) \,||\, p(z)] \tag{2.9}$$

Here, x represents input, z represents the internal latent representation and x' represents the reconstructed output. $q_\phi(z|x)$ and $p_\theta(x|z)$ represent the encoder and decoder's probability distribution, respectively. The term towards the first right represents the reconstruction error for the model while the second term towards the right is known as Kullback-Leibler (KL) divergence. KL divergence fits the latent distribution to a standard normal distribution with a mean μ of 0 and a standard deviation σ of 1.

Conditional Variational Autoencoder (CVAE)

Conditional Variational Autoencoder (CVAE) is an enhancement of the Variational Autoencoder (VAE) model discussed in the previous section. It was first introduced by Sohn *et al.* [171]. While the VAE model is a generative model that can generate all the classes from input training data, it cannot produce an output for a particular class present in training data in all its variances. This drawback is removed in the CVAE model. CVAE is a generative DL model consisting of two sub-models: Encoder (E) and Decoder (D). The encoder uses the mean and standard deviation from all the previous hidden layers to create a latent distribution such as a Gaussian distribution.

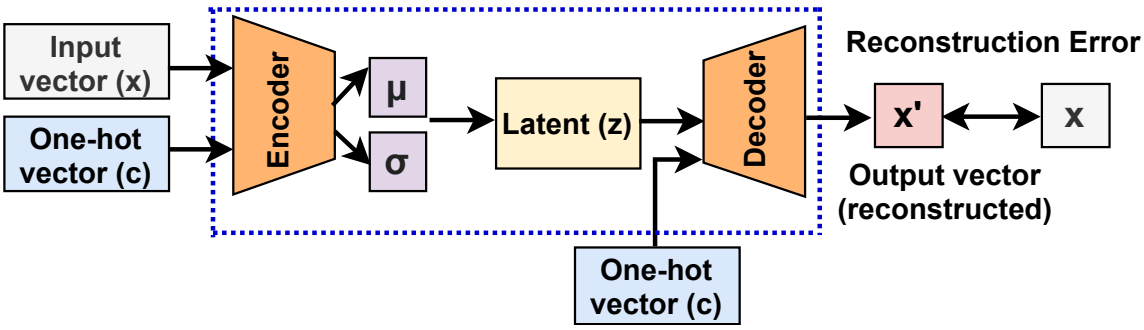


Figure 2.12: Architecture of Conditional Variational Autoencoder (CVAE)

The decoder helps to synthesize the input data from the latent variable. This model learns to generate new data as well as learn from the latent vector. Training is done using the Stochastic Gradient Variational Bayes estimator [170] which helps the encoder to learn the approximation of posterior distribution.

During training, CVAE needs a piece of extra input information in the form of a one-hot vector or conditioned vector for both the encoder and decoder for a specific input class. The encoder synthesizes latent data for a particular class by taking a random input and a one-hot vector representing that class. The one-hot vector in this case is the class label that aids in the regularization of the output from the decoder which can synthesize diverse varieties of the specific input class from the encoder's latent data. This model makes the latent space more robust to noise and dependent on the input labels. By doing this, the model can learn to group similar objects in latent space. The architecture of CVAE is similar to VAE with an additional input that represents the one-hot vector y along with the input features x for both the encoder and decoder. Fig. 2.12 shows the standard architecture of a CVAE model.

The ultimate goal of the CVAE model is to optimize the log-likelihood of the data $p(x)$ subject to an encoding error in the presence of a conditional (one-hot) variable c. The overall loss function for a CVAE is represented in eq.(2.10). A CVAE model is trained to minimize this loss.

$$L(\phi, \theta) = \mathbb{E}\left[log\, p_\theta(x|z,c)\right] - D_{KL}[q_\phi(z|x,c) \,||\, p(z|c)] \tag{2.10}$$

Here, x represents input data, z represents the encoder's latent representation. $q_\phi(z|x,c)$ represents the probability of the encoder for synthesizing latent data based on an input x and a conditioned variable c. $p_\theta(x|z,c)$ represents the probability of the decoder for synthesizing output data based on the encoder's latent data and conditioned variable. $p(z|c)$ represents the conditional probability of the encoder's latent representation. $\mathbb{E}\left[log\, p_\theta(x|z,c)\right]$ represents the reconstruction error of the CVAE and the term $D_{KL}[q_\phi(z|x,c)||p(z|c)]$ represents the regularization term or Kullback-Leibler divergence (KL divergence) between $q_\phi(z|x,c)$ and $p(z|c)$. KL divergence measures the distance between the encoder's output distribution and a standard Gaussian distribution [172].

Semi-supervised Generative Adversarial Network (SGAN)

The Semi-supervised Generative Adversarial Network or SGAN is an enhancement of a vanilla GAN model [28]. This architecture consists of simultaneous training of a generator model and two discriminator models (one supervised and another unsupervised). The supervised classification model provides better generalization on unseen or unlabeled samples and the generator model synthesizes plausible samples from its input domain. Fig. 2.13 represents the standard architecture of the SGAN model.

For semi-supervised learning, the samples synthesized by the generator model are added to the input data with N classes to create a new class for synthesized data which is represented by the $(N+1)th$ output. The discriminator, in this case, works in both supervised as well as unsupervised modes. During unsupervised training, the

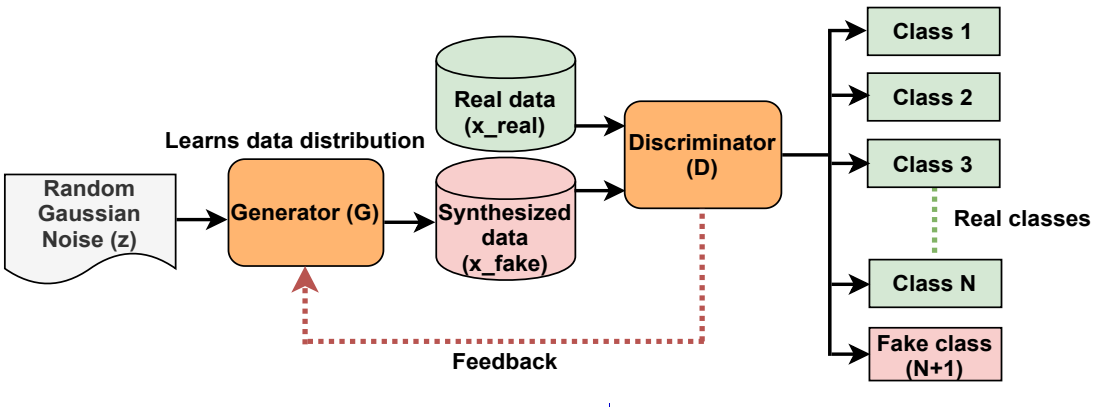


Figure 2.13: Architecture of Semi-supervised GAN (SGAN) model

discriminator is trained similar to a simple GAN model to segregate real data samples from synthesized ones. The supervised discriminator is trained to classify the labels for N classes from real data. The loss function for the SGAN generator model is given in eq.(2.7). The total loss L_{tot} for the SGAN discriminator [28] is measured as given in eq.(2.11).

$$L_{tot} = L_{super} + L_{unsuper} \tag{2.11}$$

Here, the loss of the unsupervised discriminator model $L_{unsuper}$ is given in eq.(2.6). The supervised discriminator is trained to reduce the error between the actual class labels and the predicted class labels to the minimum. The loss equation for the supervised discriminator model is given in eq.(2.12). Here, x is the input data with an N-dimensional output vector, y represents the class labels and $p_{model}(y|x)$ represents the predicted probability distribution.

$$L_{super} = \mathbb{E}_{x,y\sim p(x,y)} \left[log\, p_{model}(y|x, y < N+1) \right] \tag{2.12}$$

Chapter 3

Literature Survey

In this chapter, we provide a comprehensive summary of current literature for known/-typical, unknown, atypical, and polymorphic network attack detection using DL. For this research, the DL-based NIDS are classified as misuse-based, anomaly-based, hybrid, adversarial generation-based, and other paradigms. Furthermore, we offer an overview of ongoing research focusing on the analysis of the quality of synthetic adversarial attacks.

3.1 Misuse-based NIDS

A misuse-based NIDS, as explained earlier in Chapter 2, can observe the network for normal and abnormal traffic and match those patterns with a known set of patterns from its database. A DL model, trained using supervised learning, employs the concept of misuse-based attack detection by utilizing the existing knowledge of attacks to analyze and identify similar attack traffic [6]. DL models such as Deep Neural Network (DNN) [2,120,173–175], Convolutional Neural Network (CNN) [123,127,134, 147,176], Long Short Term Memory (LSTM) [124,129,177–179], and Gated Recurrent

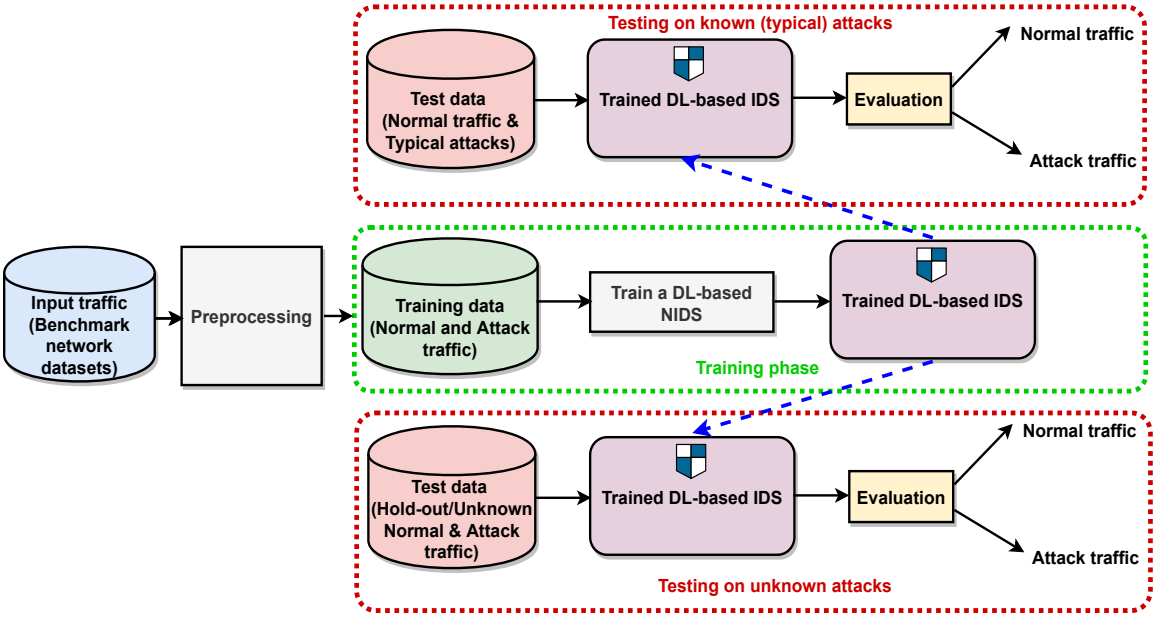


Figure 3.1: A generic workflow for misuse-based known (typical) and unknown network attack detection using DL based on current literature [4].

Unit (GRU) [14, 180, 181] have been applied to identify known/typical, unknown and atypical network attacks. In figure 3.1, we represent a general workflow for misuse-based known (typical) and unknown network attack detection using DL based on the current literature. The typical attacks are a subset of input data that the DL model has seen before. The unknown attacks are a hold-out attack traffic subset that the DL model is not exposed to during training and is kept aside for model evaluation. The DL-based NIDS applies pattern matching based on previous knowledge to identify network intrusions.

Vinayakumar *et al.* [120] employ DNN for building an efficient and intelligent intrusion detection system that can identify known attacks. Their study uses manual hyperparameter optimization to identify the best structure for DNN which is later evaluated using several benchmark datasets such as KDDCup 99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS2017. The experimental results reveal that the proposed model can effectively identify host and network-based events as compared

to other machine learning classifiers. Khare *et al.* [173] employ the Spider Monkey Optimization (SMO) technique for reducing the input dimensions before training a DNN model for intrusion detection using KDD Cup99 and NSL-KDD datasets. Their proposed model is compared with a vanilla DNN and Principal Component Analysis-based DNN (PCA-DNN) and results on known attacks signify that it performs better in terms of several commonly used evaluation metrics.

Bedi *et al.* [174] propose a 2-layered ensemble approach for improved intrusion detection. The first layer consists of three models, binary eXtreme Gradient Boosting (b-XGBoost), Siamese Neural Network, and DNN that classify attacks from benign samples. The second layer is a multi-class eXtreme Gradient Boosting that identifies multiple attack classes. The main aim of using the ensemble is to lower the misclassifications due to the class imbalance for the minority attack classes and improve the detection rate of such attacks. The evaluation is performed only for known attacks using NSL-KDD and CIDDS-001 datasets and after comparison with other classifiers, their model gives better results. Lei *et al.* [175] introduce a pruning based DNN model for attack identification. Pruning is employed on the DNN structure to reserve only important information about the attacks and reduce model complexity. The KDD Cup 99 dataset is used to evaluate the model performance in identifying attacks. Although the authors employ their technique for detecting both known and unknown attacks, their model only identifies known attacks with high detection rates as compared to unknown attacks. Additionally, the dataset employed is very old and therefore, does not represent current real-world attacks.

In [1], we focus on the detection of known and unknown DoS and DDoS attacks using deep learning-based algorithms such as DNN and LSTM. The models are trained using a more recent benchmark, the CICIDS2017 dataset. Further evaluation of these models is conducted using a synthesized dataset of unknown attacks, ANTS2019. The

model performance is then evaluated for different scenarios by training on one dataset and testing on another. It is observed that initially, the models fail to detect unknown attacks but after retraining at regular intervals, the performance gradually improves.

The authors in [123, 127, 134] analyze the performance of a Deep Convolutional Neural Network for intrusion detection using NSL-KDD, KDD99, and UNSW-NB15 datasets respectively. The proposed model is compared with several state-of-the-art models and shows higher classification accuracy against known/typical attacks. Zhang *et al.* [176], introduce a new Parallel Cross CNN (PCNN) model for improving the feature extraction and detection of minority attack flows. PCNN employs a feature fusion technique to automatically learn features from attack samples with fewer flows. The performance evaluation and comparison of PCNN show that it is successful in improving the classification of minority attacks. The evaluation of novel or unknown attack flows is left for future work. Chen *et al.* [147] propose an Evolutionary CNN (ECNN) model to classify attacks. Their technique uses the multi-objective immune algorithm for the optimization of the model. The training, evaluation, and comparison of the ECNN model are done using NSL-KDD and UNSW-NB15 datasets, and high classification accuracy on known attacks is achieved.

Xu *et al.* [182] propose a meta-learning framework based on the CNN model called as FC-Net. Their framework consists of two important building blocks for feature extraction and comparison. The model learns feature maps from input traffic and then compares these feature maps for classification into different categories. Two different datasets, ISCX2012 and CICIDS2017 are employed for this research. Training is done using one dataset and evaluation using another. Results indicate that the proposed technique achieves better detection rates on known attack samples as well as unknown samples from the untrained dataset. Ho *et al.* [183] propose an IDS based on a CNN model. Their IDS is trained and evaluated for known and unknown attacks using

the CICIDS2017 dataset. To evaluate the model on unknown attacks, the authors trained four different versions of the same model with four different combinations of DoS attacks. The model versions are trained using one class of DoS attacks and evaluated on other classes. Experimental results show that the proposed model when trained on one attack class has the potential to identify other attack classes (unknown samples) too.

Althubiti *et al.* [124] employ LSTM for intrusion detection using the benchmark CIDDS-001 dataset. After comparative analysis with other ML models, the authors attained a reasonable classification accuracy on known network attacks. Lee *et al.* [129] combine an LSTM model and Random Forest (RF) model for detecting known attacks using the UNSW-NB15 dataset. The LSTM captures the sequential characteristics from input data which are then fed into the RF classifier. The model performance is compared with traditional classifiers and based on the results, the proposed model achieves the highest accuracy. Yang *et al.* [177] introduce an Attention-based LSTM for known intrusion detection using the KDD-Cup99 dataset. The Attention method allows the model to extract critical information or features from input data that are fed into the LSTM classifier. The classification accuracy of the proposed technique is improved as compared to traditional classifiers. Imrana *et al.* [178] employ a Bidirectional LSTM (BiDLSTM) for classifying known attacks using the NSL-KDD dataset. A BiDLSTM consists of two LSTMs; the first one is trained on input data while the second LSTM is trained using a reversed copy of input data to reduce the vanishing gradient problem. The proposed model improves the overall detection rate against known attacks. Kanna *et al.* [179] introduce a spatial and temporal-based integrated IDS approach to improve the classification of known attacks using multiple datasets such as NSL-KDD, ISCX-IDS2012, and UNSW-NB15. The spatial aspects of input data are learned using an Optimized CNN model, the

Hierarchical Multi-scale LSTM (HMLSTM) learns hierarchical relationships of the input data and obtains time features. The proposed model provides better accuracy and a lower false alarm rate as compared to traditional IDS.

Manavi *et al.* [180] employ a GRU to identify known attacks using KDD cup99 data by analyzing their characteristics. Furthermore, they use a Genetic algorithm (GA)-based IDS to identify any abnormalities in normal observations based on prior knowledge of attacks. Once the attacks are identified, the IDS is retrained to improve its accuracy and reduce the false alarm rate as compared to traditional ML/DL classifiers. Liu *et al.* [14] propose a bidirectional GRU classifier based on a hierarchical attention mechanism. The attention method is employed to identify feature importance using the UNSW-NB15 dataset. High classification accuracy and low false alarm rate indicate the superior performance of this model in identifying known attacks. Assis *et al.* [181] employ a GRU-based IDS for Software Defined Networks. Evaluation of their model is done using newer benchmark datasets such as CICIDS2018 and CICDDoS 2019. Further testing of the IDS is done using real network flows and high detection rates make GRU a feasible solution in their case. A mitigation approach is employed to identify and block the attacker in the network.

Although several research works have highlighted the effectiveness of a Feed-Forward DNN model in identifying network intrusions, it has a high computational cost as compared to traditional ML models. These costs can reduce the training speed and potentially result in suboptimal solutions [184]. A CNN model can automatically learn multiple characteristics from input data and therefore requires the least pre-processing time as compared to other classifiers. Although CNN requires fewer parameters and provides better performance as compared to a Feed-Forward DNN, the training time required for this model is even higher than a standard DNN. CNN can efficiently analyze spatial data but when used to analyze nonspatial data,

its performance degrades [185]. Although a CNN can identify whether a network flow is normal or malicious, it is not effective when analyzing the trajectory of the attack based on previous information. Since LSTM can handle long-term dependencies, it is widely adopted to analyze sequential network patterns. In cybersecurity, these models can predict the next state of the attack based on the previous state. This ability makes LSTM the most effective model for time series analysis [185]. However, due to the complex nature of this model, it requires more training parameters as compared to CNN, hence taking the longest time to train.

In summary, although some initial research has been done for unknown, and atypical network attack detection using misuse-based DL models, the identification of such attacks is still a challenging task since these models assume that the attacker usually launches similar types of attacks on different networks. The detection rates reported for unknown and atypical attacks are significantly low as compared to known attacks. Moreover, exhaustive training with a huge, labeled dataset, and consequent retraining with new attacks and normal traffic is needed for misuse-based DL models to identify unknown and atypical/polymorphic attacks.

In Table 3.1 and Table 3.2 we present a comparative summary of DL research in misuse-based NIDS for known, unknown, and atypical attacks.

3.2 Anomaly-based

An anomaly-based NIDS can observe the regular network traffic and any deviation from its typical behavioral pattern is marked as anomalous. When DL is applied in anomaly detection, the model is trained in an unsupervised mode only using normal network traffic since labeled attack traffic is not always available [6]. The learned DL model is then deployed to analyze and identify network anomalies. DL models such as

Table 3.1: Summary of DL research in Misuse-based NIDS for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA). Feature selection: FS, Hyperparameter optimization: HPO, Not Available: N/A.

Ref.	**Model**	**Dataset**	**FS**	**HPO**	**Results: TA**	**Results: UA**	**Results: AA/PA**
[120]	DNN	KDD'99 NSL-KDD UNSWNB15 Kyoto WSN-DS CICIDS2017	Yes	Manual	*Accuracy:* KDD'99: 92.7% NSL-KDD: 78.9% UNSW-NB15: 76.1% Kyoto: 88.5% WSN-DS: 98.2% CICIDS2017: 93.1%	N/A	N/A
[173]	DNN	KDD'99 NSL-KDD	Yes (SMO)	Manual	*Accuracy:* KDD'99: 92.8% NSL-KDD:99.4%	N/A	N/A
[174]	b-XGBoost Siamese Neural Network DNN	NSL-KDD CIDDS-001	N/A	Manual	*Accuracy:* NSL-KDD: 80% CIDDS-001: 89%	N/A	N/A
[175]	Prunning DNN	KDD'99	Yes	Manual	*Accuracy:* 99.04%	*Accuracy:* 10.5%	N/A
[1]	DNN LSTM	CICIDS2017 ANTS2019	Yes (PCA)	Manual	*Accuracy:* DNN: 95.44% LSTM: 95.53%	*Accuracy:* DNN: 98.72% LSTM: 96.15%	N/A
[134]	Deep CNN	NSL-KDD	Yes	Random Search	*AUC:* 0.926	N/A	N/A
[123]	CNN	KDD'99	N/A	Manual	*Accuracy:* 98.25%	N/A	N/A
[127]	CNN	UNSW-NB15	N/A	Grid Search	*Accuracy:* 94.4%	N/A	N/A
[176]	PCNN	CICIDS2017	Yes	Manual	*Accuracy:* 98.21%	N/A	N/A
[147]	ECNN	NSL-KDD UNSW-NB15	N/A	Evolutionary	*Accuracy:* NSL-KDD: 99.70% UNSW-NB: 89.01%	N/A	N/A
[182]	FC-net	ISCX2012 CICIDS2017	Yes	Manual	*ISCX2012:* Accuracy: 97.87% DR: 98.88% *CICIDS2017:* Accuracy: 95.39% DR: 98.19%	*ISCX2012:* Accuracy: 93.73% DR: 95.47% *CICIDS2017:* Accuracy: 94.64% DR: 99.62%	N/A
[183]	CNN	CICIDS2017	Yes	Manual	*Accuracy:* 99.64% *DR:* 99.29%	*DR:* Hulk: 87.29% Goldeneye: 57.8% Slowloris: 38.94% Slowhttp: 85.73%	N/A

Table 3.2: Summary of DL research in Misuse-based NIDS for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA) (contd.)

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[124]	LSTM	CIDDS-001	Manual	Manual	*Accuracy:* 84.83%	N/A	N/A
[129]	LSTM, RF	UNSW-NB15	Yes	Grid Search	*Accuracy:* 99.37%	N/A	N/A
[177]	Attention-LSTM	KDD'99	Yes	Manual	*Accuracy:* 94.3%	N/A	N/A
[178]	BiDLSTM	NSL-KDD	N/A	Manual	*Accuracy:* 82.05%	N/A	N/A
[179]	Opt. CNN, Hierarchical Multi-scale LSTM	NSL-KDD ISCX-IDS UNSW-NB15	Yes	Lion Swarm Optimization	*Accuracy:* NSL-KDD: 90.67% ISCX-IDS: 95.33% UNSW-NB15: 96.33%	N/A	N/A
[180]	GRU, GA	KDD'99	N/A	Manual	*Accuracy:* 99.91%	N/A	N/A
[14]	GRU	UNSW-NB15	Yes	Manual	*Accuracy:* 98.76%	N/A	N/A
[181]	GRU	CICDDoS2019 CICIDS2018	N/A	Manual	*CICDDoS2019:* Recall: 99.9% *CICIDS2018:* Recall: 93%	N/A	N/A

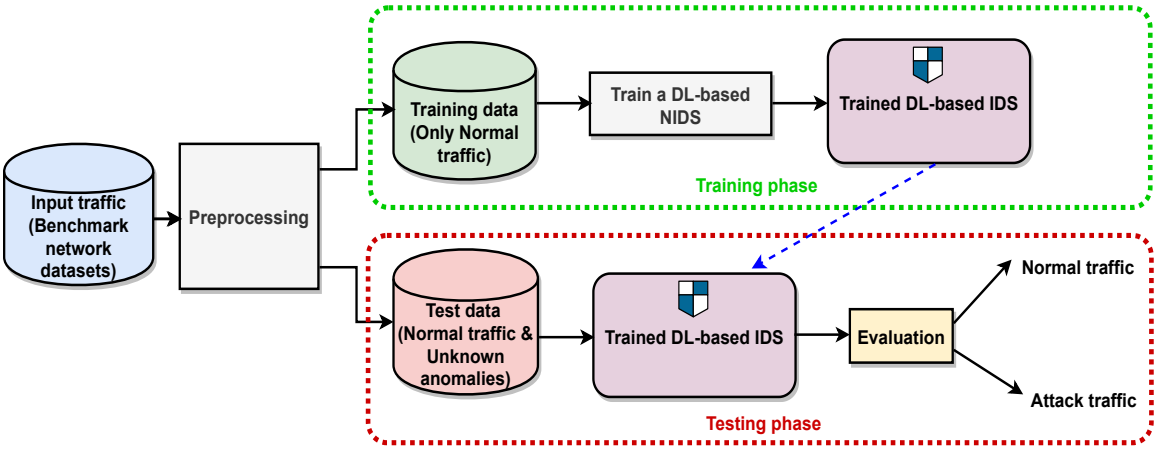


Figure 3.2: A general workflow for unknown network anomaly detection using DL [5].

Deep Belief Network (DBN), Deep Autoencoder (DAE), and Variational Autoencoder (VAE), Conditional Variational Autoencoder (CVAE) have been applied by some researchers to identify unknown network attacks or anomalies [5, 49–55, 186–190]. In figure 3.2, we represent a general workflow for unknown network anomaly detection using DL.

Zhang *et al.* [186] employ DBN in an IoT environment for intrusion detection. A Genetic Algorithm (GA) is used to find the optimal structure for a DBN which improves the generalization and reduces the complexity of the model. The evaluation of the NSL-KDD dataset signifies that the proposed GA-DBN model has a better classification accuracy as compared to other traditional models. Manimurugan *et al.* [187] propose a DBN algorithm for intrusion detection in a smart medical IoT environment. For analyzing the performance of this IDS, the CICIDS2017 dataset is used. DBN-based IDS is compared with other traditional models based on several performance metrics such as accuracy, precision, recall, F1 score, and detection rate. Evaluation of other IoT attack datasets is left for future work. Wang *et al.* [188] propose an enhanced hybrid model based on DBN, Enhanced Grey Wolf Optimizer (EGWO), and Kernel-based Extreme Learning Machine (KELM) for intrusion detection. EGWO is employed to improve the dimension reduction capability of DBN and to provide better classification results using KELM. Several benchmark datasets such as KDD '99, NSL-KDD, UNSW-NB15, and CICIDS2017 are employed for performance evaluation. The proposed algorithm has a better performance in terms of accuracy, precision, recall, true positive rate, and false positive rate as compared to existing methods.

Qureshi *et al.* [49] use a combination of a deep sparse autoencoder and self-taught learning for attack identification. The proposed system extracts features from an input dataset such as NSL-KDD. It then uses the knowledge of extracted features by merging them with actual features used for training a sparse autoencoder. When comparing their model with a Multi-Layer Perceptron (MLP) and a Deep Belief Network (DBN), their results in terms of different DL metrics show improvements in attack detection. Elsayed *et al.* [50] introduce a hybrid attack detection approach based on LSTM-Autoencoder and One-class Support Vector Machine (OC-SVM).

The LSTM-Autoencoder is trained only with normal data from the InSDN dataset and learns the latent representation of normal features. This latent information is then fed to the OC-SVM for further classification. Experimental results show that the hybrid model can identify anomalies in the network traffic effectively. Xu *et al.* [51] propose a DAE model to identify anomalies in network traffic based on the NSL-KDD dataset. Their approach utilizes an outlier analysis algorithm to identify the outliers in the training data to reduce the detection bias. Experimental results confirm the superior performance of this model as compared to other state-of-the-art in terms of accuracy, precision, recall, and F1_score.

An *et al.* [52] employ a VAE for anomaly detection based on reconstruction probability as the anomaly score. Their idea assumes that anomalous data shows a higher variance and lower reconstruction probability. Performance evaluation is done using KDD'99 and MNIST datasets. The proposed model shows better ROC-AUC values as compared to a simple Autoencoder (AE), Principal Component Analysis (PCA), and Kernel PCA. Choi *et al.* [53] compare the performances of different variations of DAE for attack identification. DAEs are trained using normal data in an unsupervised manner. Attacks are detected by measuring the anomaly score using reconstruction error at a specific threshold. Stacked AE and VAE models performed the best in their case. Nguyen *et al.* [189] propose a hybrid network attack identification approach based on an unsupervised VAE for anomaly detection and a gradient-based fingerprinting technique for explaining and verifying the identified anomalies. The evaluation is done using the UGR dataset which demonstrates the superior performance of this proposed model as compared to other models such as DAE, VAE, and Gaussian Based Thresholding (GBT) in terms of Receiver Operating Characteristic (ROC) curves. Zavrak *et al.* [5] compare the performance of classifiers such as DAE, VAE, and One-Class SVM (OCSVM) to identify unknown attacks or anomalies from

network flow features. The experimental analysis is carried out using the CICIDS2017 dataset and it is observed that the attack detection rate of VAE is better as compared to the other observed models.

Lopez-Martin *et al.* [54] introduce a framework ID-CVAE for intrusion detection systems in an IoT network. The proposed technique can reconstruct and restore missing features for training datasets to improve classification accuracy. The unsupervised classification of attacks is done by measuring the distance between the target label and prediction using reconstruction error. Experimental results demonstrate that the proposed technique gives better classification results by training on the NSL-KDD dataset as compared to other well-known ML models. Hannan *et al.* [55] propose an anomaly detection system based on a CVAE model. Their technique employs the latent representation from the encoder to differentiate between anomalous and normal traffic using bimodal distribution. The technique is studied using NSL-KDD and CICIDS2017 datasets and provides better F1 scores as compared to other classification methods such as Support Vector Machine (SVM), Decision Tree (DT), DAE, and VAE. Although the paper states that the proposed model can improve the detection of unknown attacks, the experiments are not detailed, and no information is provided about the number of false alarms generated.

Yang *et al.* [190] propose a hybrid two-stage learning technique for the detection of known and unknown attacks using CVAE and Extreme Value Theory (EVT). The first stage of the proposed technique aims to investigate a measure to minimize the misclassification of known anomalies whereas the second stage aims to explore a measure to minimize the misclassification risk of inferring unknown anomalies. EVT examines the high reconstruction errors for identifying unknown attacks in the second phase. A clustering technique is further employed to examine normal traffic. Experimental evaluation is done using NSL-KDD and CICIDS2017 datasets. Although the

results show that the CVAE-EVT technique has a better performance as compared to other algorithms, the TPRs for unknown attacks are still low and need further improvements.

In summary, these research works demonstrate that DBM, DAE, VAE, and CVAE can be successfully used for network anomaly detection. However, since these models employ unsupervised learning, their performance is affected by high false alarm rates [5]. Most notably and given the typically large variation in network traffic, the possibility that unsupervised models may discard legitimate benign traffic as suspect or potentially malicious is high, leading to a negative impact on the Quality of Service. In a real network, the number of anomalous data observations is much smaller than normal data observations which makes this network traffic highly imbalanced. It is further difficult to set a generic threshold function to define the boundary between normal and anomalous observations for identifying unknown, and dynamically changing atypical/polymorphic attacks. If the threshold function for the normal class is set too wide, the accuracy of detection is reduced. On the contrary, if the threshold function for the normal class is set too narrow, the false positive rate will increase [6]. Additionally, the training for DAE and its variants is based on gradient-based optimization and approximation which can introduce some loss for these models [191].

Table 3.3 and Table 3.4 present a comparative summary of DL research in anomaly-based NIDS for unknown and polymorphic attacks.

3.3 Hybrid

A hybrid NIDS combines the best characteristics of misuse-based and anomaly-based NIDS. DL models are trained using hybrid techniques to classify known/typical attacks and to analyze and identify unknown attacks. In this section, we discuss several

Table 3.3: Summary of DL research in Anomaly-based NIDS for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA).

Ref.	**Model**	**Dataset**	**FS**	**HPO**	**Results: TA**	**Results: UA**	**Results: AA/PA**
[186]	GA, DBN	NSL-KDD	N/A	Yes (GA)	*Accuracy:* DoS: 99.45% R2L: 97.78% Probe: 99.37% U2R: 98.68%	N/A	N/A
[187]	DBN	CICIDS2017	Yes (in-built)	N/A	*Accuracy:* Normal: 99.37% Botnet: 97.93% Brute Force: 97.71% DoS/DDoS: 96.67% Infiltration: 96.37% Port scan: 97.71% Web attack: 98.37%	N/A	N/A
[188]	DBN, EGWO, KELM	KDD'99 NSL-KDD UNSW-NB15 CICIDS2017	Yes (DBN)	EGWO	*Accuracy:* KDD'99: 98.6% NSL-KDD: 98.6% UNSW-NB15: 93.42% CICIDS2017: 97.15%	N/A	N/A
[49]	DAE	NSL-KDD	Yes (in-built)	Manual	N/A	*Accuracy:* 84.60%	N/A
[50]	LSTM-AE, OC-SVM	InSDN	Yes (DAE)	Manual	N/A	*Accuracy:* 90.5%	N/A
[51]	DAE	NSL-KDD	Yes (in-built)	Manual	N/A	*Accuracy:* 90.61%	N/A
[52]	VAE	KDD'99	Yes (in-built)	Manual	N/A	*AUC ROC:* DoS: 0.795 R2L: 0.777 U2R: 0.782 Probe: 0.944	N/A
[53]	AE Denoising AE Stacked AE VAE	NSL-KDD	Yes (in-built)	Manual	N/A	*Accuracy:* AE: 91.7% Denoising AE: 88.02% Stacked AE: 87.82% VAE: 87.66%	N/A
[189]	VAE, Gradient-based fingerprinting	UGR	Yes (in-built)	Manual	N/A	*AUC ROC:* 0.947	N/A
[5]	VAE	CICDS2017	Yes (in-built)	Manual	N/A	*AUC ROC:* 0.7596	N/A
[54]	CVAE	NSL-KDD	Yes (in-built)	Manual	N/A	*Accuracy:* 80.10%	N/A

hybrid DL-based research works employed in known and unknown network intrusion detection [192–198].

Papamartzivanos *et al.* [192] introduce an autonomous NIDS based on DL. The

Table 3.4: Summary of DL research in Anomaly-based NIDS for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA) (contd.)

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[55]	CVAE	NSL-KDD CICIDS2017	Yes (in-built)	Manual	N/A	*F1_score:* NSL-KDD: 88.44% CICIDS2017: 88.89%	N/A
[190]	CVAE, EVT	NSL-KDD CICIDS2017	Yes (in-built)	Manual	N/A	*NSLKDD1:* Recall: 54.17% *NSLKDD2:* Recall: 51.03% *CICIDS1:* Recall:49.12% *CICIDS2:* Recall: 59.83%	N/A

authors claim that their technique is adaptive and can achieve high attack detection in dynamic environments using Self-Taught Learning (STL) and MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) methodology. This is achieved using sensors/monitors that continuously sense the network activity and actuators that take actions. The authors employ network sniffers to capture network traffic and then use the network analyzer to transform raw traffic into network flows which are then fed into the supervised IDS for attack detection. The unlabeled network flows are fed into the adaptive STL module to update the IDS. The supervised IDS is trained using KDDCUP'99 and NSL-KDD datasets. An unsupervised Sparse AE is utilized to extract information from unlabeled data which is further sent to the supervised FeedFoward AE for multi-class attack detection. The authors have further compared the performances of static and adaptive IDS and confirmed that their technique can improve the detection rates of the IDS in a dynamic environment. However, in real-world networks, the incoming data can belong to different probability distributions which pose a challenge to the IDS assuming similar probability distribution for known and unknown attacks.

Zhang *et al.* [193] propose a NIDS based on Open-set Classification Network (OCN) to identify unknown attacks. OCN can classify an observed sample into a known attack/normal class or a sole unknown class. Their hybrid NIDS constitutes a CNN-based classifier and a semantic embedding clustering method. The supervised CNN classifier identifies known attacks by training on KDDCUP'99 and CICIDS2017 datasets. Unknown attacks are detected using the OCN. Further classification of unknown attacks is done using an unsupervised semantic embedding clustering technique which consists of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and K-means algorithms. While DBSCAN can identify several hidden unknown attacks, the K-means algorithm is employed to cluster semantic embeddings of multiple instances of unknown attacks identified by the OCN. The CNN classifier is then incrementally updated using centroids of clusters of unknown attacks identified in the previous step. Several attack classes from the benchmark datasets are set aside to act as unknown attack classes during the test phase. Extensive experiments are conducted using a single class of unknown attack as well as multiple classes of unknown attacks from the datasets. The results from these experiments indicate the effectiveness of this approach against state-of-the-art techniques. However, when using OCN, an identified unknown class can have both normal and attack traffic, and classifying these observations needs manual intervention.

Ahmad *et al.* [194] introduce a DL-based ensemble technique for identifying unknown network attacks. Their proposed ensemble consists of three DL models: AE, CNN, and LSTM. Four different benchmark datasets BoT-IoT, N_BaIoT, CICIDS2017, and NSL-KDD are employed for different scenarios of model training. This work uses dimension reduction techniques Principal Component Analysis (PCA), and Independent Component Analysis (ICA) to reduce the dimensions of the training datasets to conserve the correlation between features to the maximum. The authors

set aside some attack instances from the datasets for testing such that there is no overlap of these attacks in the training and test datasets. This ensures that the test attacks remain unknown to the hybrid DL model. Although the experimental results show that the hybrid model has a reasonable performance in identifying known and unknown attacks; the model can classify an attack of a specific category as an attack but cannot define the exact attack class. Additionally, dimension reduction techniques such as PCA and ICA can change the meaning and ranges of network features and make them difficult to interpret.

Shieh *et al.* [195] introduce an IDS that can detect unknown DDoS attacks using a hybrid framework consisting of a Bi-Directional Long Short-Term Memory (BiLSTM), a Gaussian Mixture Model (GMM), and incremental learning approach. The BiLSTM model employs supervised learning to identify normal and DDoS attack traffic. The unsupervised GMM model is used to analyze new attacks or new normal traffic that the BiLSTM classifier has not seen before. GMM employs the Gaussian probability density function to model data into clusters using a threshold value. The data marked as unknown by the GMM model is sent for labeling to the network engineer. This labeled data is then fed incrementally to both BiLSTM and GMM models for further training. The authors employ CIC-IDS2017 and CIC-DDoS2019 datasets for the training and evaluation phases. The results indicate a satisfactory performance in identifying unknown attacks. However, several problems need attention first. For example, more datasets should be employed to analyze model performance for better understanding. Manual labeling of new observations adds extra overhead to this system.

Du *et al.* [196] propose a framework named XFinder to identify unknown network anomalies. The authors follow a distributed ML approach wherein each node in the network has a DL-based IDS named XFinder which is connected to a central

server responsible for aggregating information from all the nodes and making global updates while training individual localized DL models. Each hybrid DL-based IDS (constituting a CNN and LSTM) goes through an initial training phase to learn the known normal and attack network traffic and provide classification results. During the unknown traffic detection, a distance distribution matrix consisting of features and class prototypes is created. A threshold value is configured based on the mean distance distribution. This threshold value is then applied to any unknown traffic to identify known and unknown classes. If unknown traffic is identified, it is then classified and automatically labeled using the K-means clustering algorithm. The authors employ a buffer that stores a labeled unknown class and any new classes detected are compared against the classes stored in the buffer. A network updater incrementally updates the IDS with labeled unknown traffic to improve its classification. Three different benchmark datasets namely KDD Cup'99, UNSW-NB15, and CICIDS2017 are used for their experiments. Some classes of attacks are set aside as unknown attacks to evaluate the model against anomalies. While the average accuracy of the model for unknown anomalies increased using this technique, the inclusion of a buffer to store network anomalies adds additional overhead to the system. Moreover, this paper does not provide any comparative analysis with other state-of-the-art techniques for anomaly detection.

Hu *et al.* [197] propose an Open-Set Recognition based IDS scheme for classifying known and unknown network attacks. Their hybrid model consists of a pretraining module which is a combination of a CNN model and a transformer encoder model. The pretraining module uses the common DL approach for examining the input data to learn the underlying characteristics of the unlabeled network traffic. The closed-set training using the benchmark dataset ISCXVPN2016 is done by employing an individual model as well as an ensemble of models. Each model consists of a CNN

module, a transformer encoder, and a Dense classifier module. The authors select 5 input classes as unknown classes of data, and 8 classes of data are selected as known classes such that there is no overlap between known and unknown classes of network traffic. During closed-set training, the traffic from the pre-trained module is fed into the classifier module for the detection of results on known classes. During the open-set test phase, the model is subjected to unknown class detection. Experimental results indicate that their approach outperforms other state-of-the-art. However, their method can only identify unknown traffic but is not able to indicate if it is an attack and further classifies that attack based on its characteristics. The ensemble of encoders in their model may improve model performance but will also add extra time overhead which is not suitable for dynamic networks.

Shieh *et al.* [198] employ a hybrid IDS based on a One-dimensional Deep Hierarchical Reconstruction Network (DHRNet) and One-Class SVM (OCSVM) to identify unknown attacks using CICIDS2017 and CICDDoS2019 datasets. The 1D-DHRNet comprises a CNN-based encoder and decoder network used for feature learning and identification of DDoS attacks from normal traffic. A Spatial Location Constraint Prototype Loss (SLCPL) loss function is employed which calculates the intra-class distance for the classification of known and unknown traffic. Further, OCSVM is employed with a certain threshold value to improve the identification of unknown and known traffic. One subset of the benchmark datasets is used for training while the other subset is kept aside to mimic unknown attacks. Evaluation results show that the proposed framework is successful in identifying unknown attacks after incremental learning. However, due to the complex structure of this IDS model, the time complexity is high which requires intervention when such a model is subjected to rapidly changing attacks.

In a nutshell, DL models using misuse-based intrusion detection match the ab-

normal events in the network with known attack patterns that they are exposed to during training. Such systems have higher detection rates and lower false positive rates for known attacks, yet detection of unknown and atypical/polymorphic attacks is possible to some extent with rigorous training iterations. On the other hand, DL models using anomaly-based intrusion detection are trained using only normal data, and anything other than normal is considered an anomaly. Such systems can detect unknown, atypical/polymorphic attacks, but suffer from increased false alarms which affect the detection rate significantly. The hybrid DL models are designed to combine the unknown, atypical/polymorphic attack detection capability of anomaly-based NIDS and the higher detection rate and trustworthiness of misuse-based NIDS. However, designing an efficient hybrid DL-based NIDS is based on the proper selection and integration of misuse-based and anomaly-based NIDS based on the application. It also depends on maintaining a balance between an increased detection rate and reduced false positives to improve the identification of unknown, atypical/polymorphic attacks [6]. Additionally, the DL-based hybrid NIDS has a complex architecture due to the addition of multiple components which add extra overhead during training and deployment. Figure 3.3 shows different categories of DL-based hybrid NIDS. The misuse-based and anomaly-based IDS are integrated in different sequences to create a hybrid system for attack detection [6].

Table 3.5 and Table 3.6 present a comparative summary of DL research in hybrid NIDS for unknown, atypical, and polymorphic attacks.

3.4 Adversarial Generation-based

Several adversarial generation-based DL models such as Generative Adversarial Network (GAN), Adversarial Autoencoder (AAE), Conditional Variational Autoencoder

Table 3.5: Summary of DL research in Hybrid NIDS for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA).

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[192]	Sparse AE, Feed Forward AE	KDDCup'99 NSL-KDD	Yes (in-built)	Manual	N/A	*DR:* 73.37%	N/A
[193]	CNN, DBSCAN and K-means	KDDCup'99 CICIDS2017	Yes (in-built)	Manual	*KDDCup'99 (Acc)*: UA=2, 86.7% UA=3, 86.7% UA=4, 86.7% *CICIDS2017 (Acc)*: UA=2, 92.1% UA=3, 90.9% UA=4, 90.8%	*KDDCup'99 (Acc):* UA=2, 90.6% UA=3, 91.8% UA=4, 91.8% *CICIDS2017 (Acc)*: UA=2, 97.5% UA=3, 92.9% UA=4, 92.7%	N/A
[194]	AE, CNN and LSTM Ensemble	BoT-IoT N_BaIoT CICIDS2017 NSL-KDD	Yes (PCA, ICA, in-built)	Manual	N/A	*BoT-IoT (Acc):* PCA: 0.66% ICA: 22.24% Full dataset: 0.41% *N_BaIoT (Acc):* PCA: 99.9% ICA: 99.9% Full dataset: 99.9% *CICIDS2017 (Acc):* PCA: 0% ICA: 0% Full dataset: 0% *NSL-KDD (Acc):* PCA: 31.31% ICA: 50.31% Full dataset: 23.21%	N/A
[195]	BiLSTM, GMM	CICIDS2017 CICDDoS2019	Yes (in-built)	Manual	*CICIDS2017 (Acc):* Wednesday: 94.2%	*CICIDS2017 (Acc):* Friday: 98.2% *CICDDoS2019 (Acc):* NetBIOS: 98% NTP: 92.3% LDAP: 95.3%	N/A
[196]	CNN, LSTM, K-means	KDD Cup'99 UNSW-NB15 CICIDS2017	Yes (in-built)	Manual	*Accuracy:* KDD Cup'99: 99% UNSW-NB15: 99% CICIDS2017: 99%	*Accuracy:* DoS Hulk 72.67% Exploits: 78.65% Reconnaissance: 79.14% DoS: 83.61% Fuzzers: 66.50%	N/A

(CVAE), and their variants are applied in cybersecurity research for launching known attacks and unknown adversarial network attacks against the IDS [7,44,57–68]. Addi-

Table 3.6: Summary of DL research in Hybrid NIDS for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA) (contd.)

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[197]	CNN, Encoder, and FC-Classifier	ISCXVPN2016	Yes (in-built)	Manual	N/A	*2-class Acc:* 79%	N/A
[198]	DHRNet OCSVM	CICIDS2017 CICDDoS2019	Yes (in-built)	Manual	*CICIDS2017 Wednesday:* Accuracy: 99.99%	*CICIDS2017 Friday:* Accuracy: 99.86% *CICDDoS2019 (Acc):* LDAP: 99.94% MSSQL: 99.90% DNS: 99.9% NetBIOS: 99.84% NTP: 99.36% UDP: 99.92% SNMP: 99.90% SSDP: 93.47% SYN: 99.89%	N/A

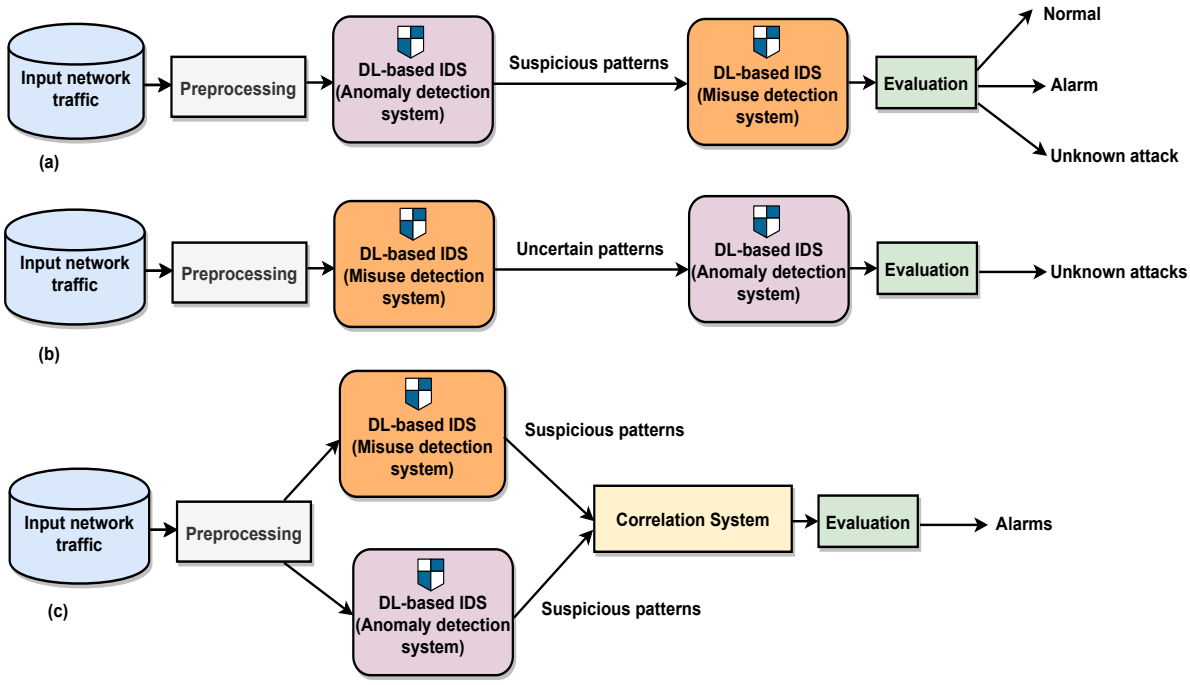


Figure 3.3: Different categories of DL-based hybrid NIDS. (a) Anomaly-misuse system, (b) Misuse-anomaly system, and (c) Parallel system [6].

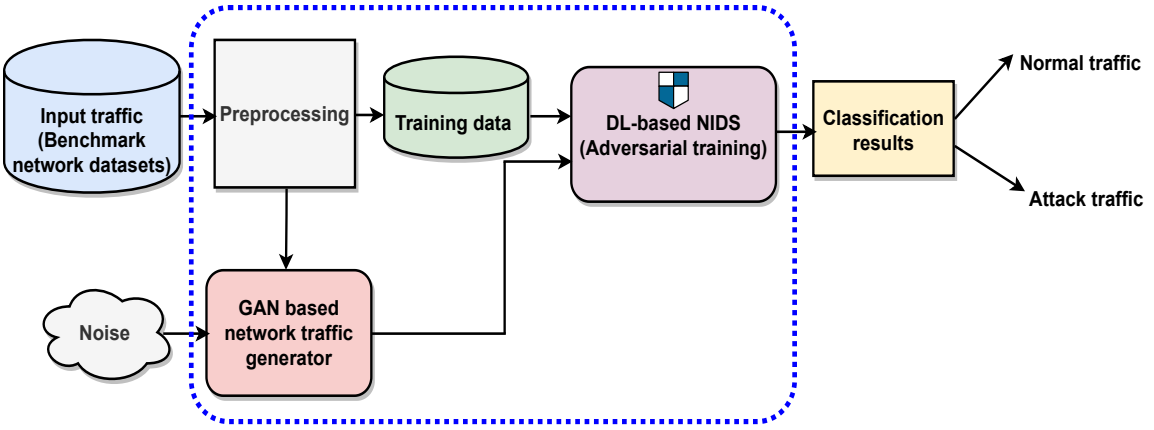


Figure 3.4: The general idea behind adversarial learning for DL-based NIDS using a generative model such as GAN [7].

tionally, a few research works also focus on using adversarial DL models for generating polymorphic network attacks against NIDS [71,199]. Figure 3.4 represents the general idea behind adversarial learning for DL-based NIDS using GAN.

Usama *et al.* [7] employ a GAN-based attacker to generate adversarial attacks that can successively evade detection by the IDS. Training of the GAN model and other IDS models such as DNN, Logistic regression (LR), Support vector machine (SVM), k-nearest neighbor (KNN), Naive Bayes (NB), Random Forest (RF), Decision Tree (DT), and Gradient Boosting (GB) techniques is done using the KDD'99 dataset. The authors show improvements in IDS performances in terms of accuracy, precision, recall, and F1 score after the adversarial training phase.

Wu *et al.* [57] employ a hybrid Deep Convolutional Generative Adversarial Network (DCGAN) to build an intelligent IDS that can identify a wide variety of attacks including adversarial attacks for an IoT network. Two new benchmark datasets named CSE-CIC-IDS2018 and CIC-DDOS2019 are employed for performance evaluation with the Fuzzy Rough Set feature selection technique which analyzes and removes redundant features. Although compared with other IDS techniques, DCGAN provides the highest overall accuracy, for edge IoT networks a lightweight and

precise IDS needs to be investigated further.

Chauhan *et al.* [71, 199] employ a GAN model to synthesize adversarial polymorphic DDoS attacks which can bypass a NIDS. To update the feature profile for launching new polymorphic attacks, the authors employ manual feature updating by changing the number of features for training and swapping unused features with used features during each new attack phase. Training is done using the CICIDS2017 dataset and experiments show a high success rate of the attack against the ML-based IDS during the initial polymorphic attack phase, improvement in detection rate during the adversarial training phase, and a significant reduction in the detection rate after another polymorphic attack phase. Although the adversarial learning phase shows improvement in the attack detection rates, the manual updating of feature profiles to launch polymorphic attacks is not always a feasible approach and needs improvements using automated techniques.

Liu *et al.* [58] propose a method to detect unknown adversarial attacks by using a Generative Adversarial Cooperative Network (GACN). The proposed approach employs GACN for synthesizing unknown adversarial network attacks to deal with data insufficiency issues during training. The attacks are synthesized based on intra categories in the embedding space. This ensures that the attacks generated are unique as compared to the attacks seen during training. The authors through this research intend to improve the IDS detection capability against unknown adversarial attacks in real-world scenarios when attack data is rare. The k-means clustering method is used to cluster unlabeled attack samples synthesized by GACN. The clustered data is then augmented with the training data and identified using the deep neural network IDS. For experimental evaluation, datasets such as CICIDS2017, CICIDS2018, and Fashion-MNIST are employed. Although experiments show improved TPRs and reduced FNRs for unknown adversarial attacks, the complexity of training and hyper-

parameter optimization is not considered by the authors. There is also a need to make the IDS more scalable in terms of attack scenarios, different network environments, and the identification of multiple classes of attacks.

Zhou *et al.* [59] introduce an IDS based on GAN and Genetic Algorithm (GA) against unknown network attack detection. This research aims to generate attack patterns from known attacks by mutating certain features to evaluate against unknown attacks and reduce the class imbalance in the training data. For classifying attacks, the predefined DL-based ResNet algorithm is applied. For generating new attacks, the authors propose a Generating Evolution Algorithm (GEA) which consists of a GAN model and a Genetic Algorithm (GA). The authors employ a BiGAN to generate adversarial data which is then fed into the GA to synthesize new adversarial attack data which acts as unknown attacks. Extensive experiments are carried out using the CICIDS2018 dataset. Samples of the Hulk DoS attack and SlowHttpTest attack are selected for training data, and samples of the GoldenEye DoS attack and Slowloris DoS attack are selected as unknown attacks for test data. The IDS is trained after class balancing using GAN and GA. Although the accuracy of unknown attacks improves with their technique, the authors have not discussed the problem of how mutating attack features can create a realistic network attack.

Although the GAN model is quite popular in cybersecurity research for adversarial training and handling class imbalance by generating better quality attack samples, it suffers from complex training convergence and mode collapse problems [200]. In mode collapse, the generator cannot fully capture the diversity in the real data distribution and cannot produce varied data samples [200].

Martin *et al.* [60] apply VAE for synthesizing data of different classes that resemble input data using the labels of the target classes. The paper aims to reduce the poor performance of other machine learning classifiers such as RF, LR, Linear SVM, and

Multi-Layer Perceptron (MLP) caused by class imbalance in the training data. After comparison, results indicate that the classifiers trained with their synthesized data perform better than the ones trained with data generated by other oversampling algorithms. Vu *et al.* [61] introduce a Multi-distributed VAE (MVAE) for improving network attack detection. In MVAE, the class label information is added to the loss term which allows the segregating of the latent space regions for different classes making them easily detectable. The MVAE synthesized samples are used to improve the classification accuracy of several ML classifiers such as Gaussian Naive Bayes (GNB), SVM, DT, and RF. The performance evaluation is done using NSL-KDD and UNSW-NB15 datasets. Experimental results indicate that the proposed MVAE model significantly improves the detection rate of other classifiers as compared to the original algorithms.

Yang *et al.* [62] introduce a hybrid model for network intrusion detection based on Supervised Adversarial Variational Autoencoder with Regularization (SAVAER) and DNN. The SAVAER is based on an enhancement of a vanilla GAN model named Wasserstein GAN with Gradient Penalty (WGAN-GP). WGAN-GP was introduced to reduce the shortcomings of a traditional GAN such as difficulty in training convergence and mode collapse problem by penalizing the norm of the gradients of the discriminator network [201]. The SAVAER model is employed to synthesize minority attack samples to balance the classes in the training dataset. The synthesized attacks are then augmented to the original training data to train a DNN classifier for identifying less frequent and unknown attacks. Based on the experiments, the proposed model outperforms other well-known class balancing techniques and classification models in terms of detection rate, F1 score, accuracy, and FPR.

VAE is employed for analyzing network flows, understanding the underlying feature relationships, and synthesizing adversarial data to improve the performance of

other classifiers against minority attacks [60–62]. However, because of the noisy inputs and improper reconstruction, the samples generated by VAE are not of high quality as compared to CVAE and GAN [202]. Therefore, it is more popularly employed for feature pre-training and dimension reduction phases for other AI classifiers [191].

Hara *et al.* [63] employ a semi-supervised AAE model that combines a VAE and a GAN model for intrusion detection using the NSL-KDD dataset. A comparison of accuracy for a semi-supervised AAE-based IDS is done against a supervised DNN-based IDS at different labeling rates. Results indicate that at a labeling rate of 0.1% or higher, the accuracy of AAE is better as compared to DNN and the misdetection rate of DNN is higher. Comparison with other state-of-the-art techniques indicates that the proposed method has better detection accuracy using only 1% and 10% labeled data instances. Aloul *et al.* [64] employ a hybrid IDS model based on AAE and KNN for improving intrusion detection on the IoT edge routers using the NSL-KDD benchmark dataset. Initially, the Synthetic Minority Over-sampling Technique (SMOTE) is used to remove any class balancing issues in the training data. AAE is used to reduce the size of the input dimensions from 41 to 16. The latent space information from the encoder in AAE is then fed into the classifier KNN for attack detection. The proposed technique surpasses other state-of-the-art models in the literature in terms of accuracy, precision, recall, and F1 score.

Abdalgawad *et al.* [65], use deep generative models such as AAE and Bidirectional Generative Adversarial Networks (BiGAN) for improving the performance of an IDS against known and unknown network attacks using IoT-23 dataset. The correlation coefficient is employed for feature selection. SMOTE and Random Undersampling techniques are explored to remove the class imbalance in the input dataset. Attacks are detected by encoding the input data to a reduced latent representation using AAE or BiGAN and feeding this information to a classifier such as KNN for attack

identification. After training, the hybrid models AAE-KNN and BiGAN-KNN provide better performance on multiple attack classes as compared to standalone models such as KNN and RF. To evaluate the performance of BiGAN on unknown attacks, synthetic attacks are created by randomly selecting a feature subset and randomly changing the values of these features. Although F1_score results indicate that BiGAN is effective in identifying unknown attack patterns even when the feature mutations are increased, performing manual changes in feature values is not always a feasible approach and some automatic techniques need to be investigated.

AAE was introduced to overcome the drawbacks of standalone GAN and VAE models [26]. Cybersecurity researchers have employed AAE for adversarial training with other classifiers to improve attack detection performance. Although AAE is a flexible model, certain training issues such as the balance between the training of the discriminator and generator which causes the gradients to explode unpredictably need further investigation.

Yang *et al.* [66] propose a hybrid IDS based on improved CVAE and DNN models. CVAE is employed to learn the sparse representations among the feature variables and target variables to synthesize new attack samples for improving the diversity in the training data. CVAE also helps in fine-tuning during the training of the DNN classifier to achieve better optimization. The datasets NSL-KDD and UNSW-NB15 are used for training and evaluation. The hybrid model is compared with several oversampling methods such as Random Oversampling (ROS), Synthetic Minority Oversampling Technique (SMOTE), and Adaptive Synthetic Sampling (ADASYN) and shows better detection results on minority attacks as well as unknown attacks. When compared against several state-of-the-art techniques, the proposed model shows a better detection rate, accuracy, and lower false positives.

Azmin *et al.* [67] propose a NIDS based on a combination of two models includ-

ing Conditional Variational Laplace Autoencoder (CVLAE) and DNN. The CVLAE is a variant of a VAE model that utilizes Laplace Approximation on the posterior distribution to enhance its expressive power by reducing errors. CVLAE model is employed to learn the latent representations from the input data and generate new data with similar characteristics. The DNN model is trained in a supervised mode using the NSL-KDD benchmark and CVLAE generated data to classify attacks from normal observations. Through experimentation, the authors show that the proposed technique shows better precision results, especially on minority attack samples when compared with other commonly used DL models.

Xu *et al.* [68] introduce a hybrid method for intrusion detection consisting of a Log-cosh Conditional Variational Autoencoder (LCVAE) and a CNN classifier. LCVAE model is an enhancement of a simple CVAE model in terms of the loss function which is improved by proposing a log hyperbolic cosine function. The new loss function is more effective in the reconstruction of input data and in diversifying minority attack samples in the training dataset. A CNN classifier is employed for feature extraction and identification of attack and normal samples. Extensive experimental results using NSL-KDD show the superior detection capability of the proposed model against other state-of-the-art techniques.

As compared to a standard VAE, CVAE is more powerful for improving the performance of other DL classifiers against zero-day attacks since it provides better regeneration results using the conditioned attribute [67]. However, CVAE is not as popular as GAN for adversarial training in network security, since GAN generates better quality samples as compared to CVAE [203]. On the other hand, CVAE is much easier to train and does not suffer from the problem of mode collapse when compared to GAN.

Adversarial generation-based DL models have been used to overcome the limi-

tations of DL-based attacks in detecting atypical and polymorphic network attacks. Although these adversarial generation-based NIDS show promising results, they experience some limitations such as the selection of an application-specific adversarial generation model, enhanced computational needs for training the model, the number and magnitude of adversarial perturbations, consistency of adversarial network attacks, and identifying formal network constraints for these attacks [204, 205].

Table 3.7, Table 3.8, and Table 3.9 provide a comparative analysis of NIDS research in Adversarial Deep Learning for unknown and polymorphic attacks.

3.5 Other Paradigms

3.5.1 Transfer Learning (TL)

Transfer learning is an ML/DL technique that focuses on improving the performance of a target ML/DL model in a target domain based on the knowledge transferred from a related source domain [206]. For example, in cybersecurity, the knowledge gained during training of a model on a huge dataset consisting of multiple DoS/DDoS attacks excluding Goldeneye DoS can be applied to identify Goldeneye DoS as a new DoS attack. Figure 3.5 represents a general idea behind the concept of transfer learning in network attack detection. Transferring knowledge from a pre-trained model for identifying a new task can considerably improve the performance of a model [207]. TL is used in many cybersecurity research works for improving intrusion detection for known and unknown network attacks [69, 208–211].

Zhao *et al.* [208] propose a TL-based technique for identifying unknown attacks. A heterogeneous transfer learning approach is employed to extract optimized attributes from the input training and evaluation data. The spectral transformation technique

Table 3.7: Summary of NIDS Research in Adversarial Deep Learning for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA).

Ref.	**Model**	**Dataset**	**FS**	**HPO**	**Results: TA**	**Results: UA**	**Results: AA/PA**
[7]	GAN, DNN, LR, SVM, KNN, NB, RF, DT, GB	KDD'99	N/A	Manual	*Accuracy:* DNN: 89.12% LR: 87.6% SVM: 88.49% KNN: 85.37% NB: 69.64% RF: 86.12% DT: 84.86% GB: 87.6%	*Accuracy:* DNN: 84.31% LR: 86.64% SVM: 84.31% KNN: 79.31% NB: 82.83% RF: 81.31% DT: 83.22% GB: 82.97%	N/A
[44]	GAN, DNN	NSL-KDD UNSW-NB15 CICIDS2017	Yes (CNN)	Manual	*Accuracy:* NSL-KDD: 84.45% UNSW-NB15: 82.53% CICIDS2017: 99.79%	N/A	N/A
[57]	DCGAN	CICIDS2018 CICDDOS2019	Yes (Fuzzy approach)	Manual	*Accuracy:* CICIDS2018: 95.22% CICDDOS2019: 98.62%	*Accuracy:* *CICIDS2018:* FTP Brute: 99.99% SSH brute: 99.94% XSS: 5.87% Slowloris: 38.85% SQL Injection: 10.34% Brute-Web: 23.22% *CICDDOS2019:* DDoS-NTP: 92.60% DDoS-DNS: 97.99% DDoS-SSDP: 99.03% TFTP: 99.86% WebDDoS: 24.24% DDoS-SNMP: 99.99%	N/A
[199]	GAN, RF	CICIDS2017	Yes (SHAP)	Manual	*Accuracy:* 94.33%	N/A	*DR:* 79.86%
[71]	WGAN, RF	CICIDS2017	Yes (SHAP)	Manual	N/A	N/A	*TPR:* 100% *TNR:* 98.6% *FScore:* 99.3%

is used to project the features on a shared latent space which reduces the difference between the distributions of train and test data to understand optimized feature

Table 3.8: Summary of NIDS Research in Adversarial Deep Learning for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA) (contd.)

Ref.	**Model**	**Dataset**	**FS**	**HPO**	**Results: TA**	**Results: UA**	**Results: AA/PA**
[58]	GACN, K-means	CICIDS2017 CICIDS2018 Fashion MNIST	Yes (in-built)	Manual	N/A	*CICIDS2018:* TPR: 93.13% FPR: 2.34% *Fashion-MNIST:* F1_Score: 99.09%	N/A
[59]	BiGAN GA ResNet	CICIDS2018	Yes (in-built)	Manual	*Precision:* 98.49% *Recall:* 96.29% *F1_score:* 97.37%	*Precision:* 90.05% *Recall:* 92.31% *F1_score:* 91.17%	N/A
[60]	VAE, RF, LR, Linear SVM, MLP	NSL-KDD	Yes (in-built)	Manual	*Accuracy:* RF: 73.61% LR: 77.29% Linear SVM: 77.23% MLP: 79.26%	N/A	N/A
[61]	MVAE, GNB, SVM, DT, RF	NSL-KDD UNSW-NB15	Yes (MVAE)	MVAE: Manual ML models: Grid Search	*AUC: NSL-KDD:* GNB: 0.924 SVM: 0.945 DT: 0.962 RF 0.943 *UNSW-NB15:* GNB: 0.928 SVM: 0.945 DT: 0.954 RF: 0.961	N/A	N/A
[62]	SAVAER, DNN	NSL-KDD UNSW-NB15	Yes	Grid search	*Accuracy:* NSL-KDD: 89.36% UNSW-NB15: 93.01%	N/A	N/A
[63]	AAE	NSL-KDD	Yes (in-built)	Manual	N/A	*Accuracy:* 82.78% *FNR:* 20% *FPR:* 13.5%	N/A
[64]	AAE, KNN	NSL-KDD	Yes (AAE)	Manual	*Accuracy:* 99.91%	N/A	N/A
[65]	AAE, Bi-GAN, KNN	IoT-23	Yes (Pearson's Correlation)	Manual	*F1_Score:* AAE+KNN 97.43% BiGAN+KNN 97.41%	*F1_Score:* BiGAN 85%	N/A
[66]	CVAE, DNN	NSL-KDD UNSW-NB15	Yes (in-built)	Manual	*Accuracy:* NSL-KDD: 85.97% UNSW-NB15: 89.08%	N/A	N/A

Table 3.9: Summary of NIDS Research in Adversarial Deep Learning for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA) (contd.)

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[67]	CVLAE, DNN	NSL-KDD	Yes (in-built)	Manual	*Accuracy:* 76%	N/A	N/A
[68]	LCVAE, CNN	NSL-KDD	Yes (CNN)	Manual	*Accuracy:* 85.51%	N/A	N/A

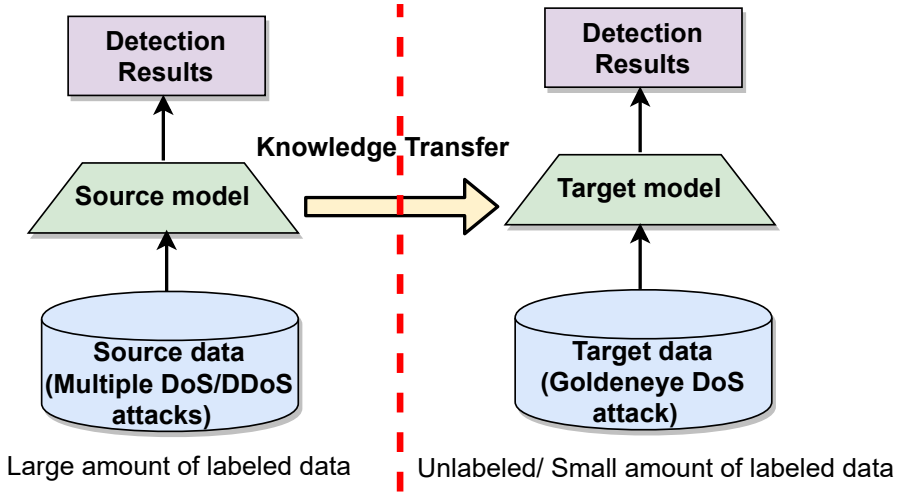


Figure 3.5: The general idea behind Transfer Learning in NIDS domain

representations. The optimized attributes from the TL module are then fed into several classifiers such as DT, NB, KNN, and SVM for performance comparison using the NSL-KDD dataset. After empirical analysis, it is concluded that the proposed classifiers trained using one attack class can effectively identify a new attack class from the same dataset as compared to the baseline classifiers.

Zhao *et al.* [209] extend their initial work [208] by finding a relationship between a known and unknown/unseen attack using a transfer learning approach based on clustering. In the proposed approach, the samples in the target domain are clustered based on clusters in the source domain. The similarity in the source and target domain clusters is identified using Euclidean distance and then the clusters in the target domain are mapped to the source domain using the K-Means clustering technique.

Experimental results using the NSL-KDD dataset show improvements in the performance of several classifiers based on this technique in detecting unknown network attacks. For [208, 209], evaluation on an unknown network attack from a different dataset, atypical and polymorphic attacks need to be explored further.

Vu *et al.* [210] introduce a TL approach based on two DAE models for intrusion detection in an IoT network. The first DAE is trained using supervised learning based on the source dataset. While the second DAE is trained using unsupervised learning on the target dataset. The knowledge from the latent representation of the first DAE is transferred to the second DAE which is then employed to identify unknown attacks in the target domain. Performance evaluation using multiple datasets generated from several IoT devices demonstrates that the proposed TL-based IDS improves the detection accuracy of unknown attacks as compared to traditional DL-based IDS. However, the training time for this IDS is higher compared to other baseline IDS. He *et al.* [211] propose a deep TL-based intrusion detection technique for detecting minority DDoS attacks in the network. The authors employ multiple DNNs for identifying one class of DDoS attacks (SYN-DDoS) in the source domain. The best DNN model is selected based on the transferability metric to identify another class of DDoS attacks (LDAP-DDoS) in the target domain. Experimental results indicate that their approach is effective in reducing the detection performance degradation of minority DDoS attacks.

Zhang *et al.* [69] propose a framework named Transferred Generating Adversarial Network-Intrusion Detection System (TGAN-IDS) to identify unknown ransomware attacks. The authors employ a dual GAN architecture with Deep Convolutional Generative Adversarial Network (DCGAN) for generating adversarial samples and TGAN for identifying between a real and synthesized sample. The generator in DCGAN is trained using only normal samples from the input dataset while the discriminator

is trained using normal and attack samples. The knowledge, structure, and initial parameters of the generator in DCGAN are transferred to TGAN to create an initial generator. The structure and parameters of the pre-trained discriminator (PreD) which is trained using normal and attack samples are transferred to the TGAN's discriminator. TGAN is trained further using unsupervised learning on normal samples. During the evaluation, TGAN is exposed to both attack and normal samples to test on unknown attacks. The discriminator of TGAN acts as an attack detector and identifies unknown attacks based on the predefined threshold value. Extensive experiments are conducted based on several data subsets for performance evaluation on unknown attacks. The ransomware attack traffic is collected from several sources while normal traffic is taken randomly from a subset of CICIDS2017 data. Further experiments are conducted with other datasets such as KDD'99, SWAT, and WADI. To represent unknown attacks, some attack samples from these datasets are set aside for evaluation and not used in training. Although results indicate the effectiveness of TGAN to identify unknown attacks, the structure of this framework is quite complex and a lightweight and less complex IDS still needs investigation.

To provide a concise overview, TL in cybersecurity is employed to improve the performance of a DL model in the target domain when there is an insufficient amount of labeled data. Until now most cyber research works based on transfer learning employ smaller datasets for training, which does not fully establish the possible advantages of TL-based ML/DL algorithms since TL requires a huge generic training dataset [212]. Although TL can speed up the process of training a model on a new task and improving its accuracy by employing pre-trained models, there are other limitations of TL such as negative transfer and overfitting [213]. A negative transfer problem can happen when the initial round of training data is way off the mark as compared to previous training data which causes the DL model performance to de-

Table 3.10: Summary of NIDS Research in TL for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA).

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[208]	DT RF NB KNN SVM	NSL-KDD	Yes	Manual	N/A	*Accuracy:* DT: 81% RF: 78% NB: 72% KNN: 78% SVM: 81%	N/A
[209]	DT RF NB KNN SVM	NSL-KDD	Yes (Information Gain)	Manual	N/A	*Accuracy:* DT: 87% RF: 68% NB: 87% KNN: 81% SVM: 84%	N/A
[210]	DAE	IoT-1 IoT-2 IoT-3 IoT-4 IoT-5 IoT-6 IoT-7 IoT-8 IoT-9	Yes (DAE)	Manual	N/A	*AUC Score:* IoT-2: 0.888 IoT-3: 0.796 IoT-4: 0.885 IoT-5: 0.943 IoT-6: 0.833 IoT-7: 0.892 IoT-8: 0.775 IoT-9: 0.743	N/A
[211]	DNN	DDoS dataset	N/A	Manual	N/A	*Detection Rate:* 99.28%	N/A
[69]	TGAN DCGAN	CICIDS2017 KDD'99 WADI SWAT	Yes (in-built)	Manual	N/A	*CICIDS2017 and Ransomeware:* Average DR: 76.18% *KDD'99:* Precision: 98.10% Recall: 99.28% F1_Score: 98.70% *WADI:* AUC: 0.98 *SWAT:* AUC: 0.86	N/A

grade [213]. Another major problem with TL is overfitting which can happen if the new DL model learns noise patterns from input data which can affect its classification performance [213]. Once these disadvantages of TL have been overcome, it is expected to rapidly improve the development of DL models in the future.

We provide a summary of NIDS research in TL for unknown, atypical, and polymorphic network attacks in Table 3.10.

3.5.2 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) is a subdivision of ML that combines the best capabilities of DL and Reinforcement Learning (RL) [214]. RL is the task of learning through trial and error with a final goal to take action. DRL adds the advantages of DL to RL which allows the agent to better understand the unstructured environment by employing larger datasets and discovering useful patterns, then taking actions to maximize the overall reward. In the DRL framework, the DNN model acts as the agent or learning system that observes the environment and performs an action based on the best learning policy. The environment then observes the state of the agent and provides feedback in the form of a reward for success and a penalty for failure based on the agent's actions.

DRL is broadly divided into two categories: Deep Q-Learning (Deep Q-Network) and Policy Learning (Policy Gradients) [215]. Deep Q-learning or Deep Quality learning is based on assessing the quality or usefulness of action in gaining a future reward. Figure 3.6 depicts a generic framework for a Deep Q-Network (DQN). DQNs employ Q-learning, also known as Value learning, which depends on selecting an action that maximizes the Q-function to infer the optimal policy. Deep Q learning only provides a discrete action space. It is represented in eq.(3.1) as given.

$$a = \underset{a}{\operatorname{argmax}}\, Q(s, a) \tag{3.1}$$

Where a represents the action, s represents the state, and $Q(.)$ represents the Q-function.

Policy Learning or Policy Gradients (PG) directly optimize the policy $\pi(s)$ that determines what action a should be taken to maximize the reward. It provides a continuous action space, for example, represented as a Gaussian distribution with an

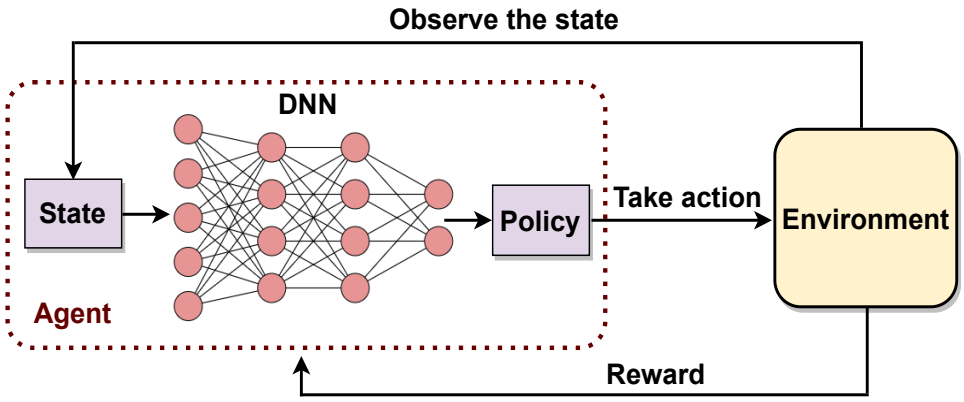


Figure 3.6: A generic framework representing Deep Q-Network (DQN)

infinite number of actions. Policy Learning is represented in eq.(3.2) as given.

$$Sample\ a \sim \pi(s) \tag{3.2}$$

DRL is employed in multiple security research works for building an adaptive IDS that can effectively identify known attacks, advanced network attacks, and zero-day attacks [216–220].

Kim *et al.* [216] employ DRL to build an online IDS based on a DAE Q-network (DAEQ-N). This research aims to achieve a maximum detection rate when identifying attacks in a real-time IDS. The authors use the OpenAI Gym framework to emulate a network environment and learn continuous network sequences for improving detection accuracy based on the actions taken and rewards received. The DAEQ-N acts as a Q-learning agent to classify attacks from normal instances. Rewards are received for each correct action such as identifying an attack as an attack while penalties are received when an attack is misclassified. Each action followed by a positive or a negative reward helps the system to learn the network better and improve the detection rates for the next iteration. However, the comparison of the proposed DAE-based Q-network with a simple DNN-based Q-network demonstrates that the proposed technique shows

more stability and improved detection results. The proposed IDS is not evaluated using any benchmark or real network datasets.

Lopez-Martin *et al.* [217] apply DRL to train their supervised intrusion detection systems for classifying network attacks and normal events using benchmark datasets such as NSL-KDD and AWID. The regular live environment of the DRL system is replaced with a pseudo-environment using a sampling function for attacks. Rewards are generated based on the classification mistakes made during the training phase. The results of several DRL algorithms such as Deep Q- Network (DQN), Double Deep Q-Network (DDQN), Policy Gradient (PG), and Actor-Critic (AC) are analyzed and compared for intrusion detection. The empirical analysis indicates that the DDQN model is the best performing model for attack detection with a lower time complexity among the other observed models as well as state-of-the-art in terms of precision, recall, F1_score, and accuracy.

Phan *et al.* [218] propose a hybrid framework to reduce the problem of dataset deficiency based on TL and Q-learning, especially for network security applications. The main aim of the proposed TL-based framework is to maximize the reward by improving the performance of the future network in the target domain. The framework is evaluated by transferring the knowledge for DDoS attack identification in a Software Defined Network (SDN) domain from a legacy network domain. The authors employ a Multilayer Perceptron (MLP) model for attack detection in both legacy and SDN networks. Several datasets such as NSL-KDD, UNSW-NB15, CICIDS2017, and CIC-DDoS2019 are employed for training the MLP IDS in the legacy network domain. The DoSinSDN dataset is used for evaluation in the SDN domain. Results indicate that by applying the proposed framework the detection rate of DDoS attacks is improved as compared to other traditional intrusion detection techniques.

Venturi *et al.* [219] employ a DRL framework to synthesize adversarial botnet at-

tacks against the current intrusion detection systems. The adversarial dataset named DReLAB is generated using two DRL models: Double Deep Q-Network (DDQN) and Deep State-action-reward-state-action (Deep Sarsa). DDQN and Deep Sarsa are trained using several botnet benchmark datasets such as CTU-13, CSE-CIC-IDS2018, and Botnet2014. The main aim of this work is to synthesize adversarial attack samples that resemble real-world attacks to assess the detection capability of current IDS against such novel attacks and apply explainability to these systems. The authors evaluate the performance of two classifiers: RF and Wide and Deep (WnD) using the DReLAB adversarial dataset. WnD is a DL framework proposed by Google for classification [221]. The experimental results determine that the adversarial attacks in the synthesized DReLAB dataset can effectively evade detection by both RF and WnD classifiers.

Sethi *et al.* [220] apply DRL for building a distributed multi-agent IDS based on DAEQ-N. Their system employs Attention mechanism to identify evolved network attacks. The encoder of the DAE uses the Attention mechanism to select the most important feature vectors with higher weights for improving the performance of the model. Empirical analysis determines that the proposed IDS is effective against several attacks from NSL-KDD and CICIDS2017 datasets as well as other adversarial and zero-day attacks as compared to other state-of-the-art techniques. The high computational resource requirement is one of the limitations of this research work. Another limitation is that it is based on the assumption that network attackers will not launch an attack on the subnetworks and gateways. In the future, these limitations need to be addressed to build an improved IDS.

Table 3.11 provides a summary of NIDS research in DRL for known, unknown, atypical, and polymorphic attacks.

To sum up, DRL-based algorithms are employed in network security to improve

Table 3.11: Summary of NIDS Research in DRL for Known/ Typical Attacks (TA), Unknown Attacks (UA), Atypical Attacks (AA), and Polymorphic Attacks (PA).

Ref.	Model	Dataset	FS	HPO	Results: TA	Results: UA	Results: AA/PA
[216]	DAEQ-N	N/A	N/A	Manual	Stability: 80%	N/A	N/A
[217]	DQN DDQN PG AC	NSL-KDD AWID	N/A	N/A	*Accuracy NSL-KDD:* DQN: 87.87% DDQN: 89.78% PG: 78.73% AC: 80.78% *Accuracy AWID:* DQN: 95.41% DDQN: 95.70% PG: 92.21% AC: 92.21%	N/A	N/A
[218]	MLP	NSL-KDD UNSW-NB15 CICIDS2017 CIC-DDoS2019 DoSinSDN	N/A	Manual	*Detection Rate:* 99%	*Detection Rate:* 99%	N/A
[219]	*Adversarial:* DDQN Deep Sarsa *IDS models:* RF WnD	CTU-13 CSE-CIC-IDS2018 Botnet2014	Yes	Manual	*F1_score:* 99%	*Detection Rate:* 88%	N/A
[220]	DAE-DQN	NSL-KDD CICIDS2017	N/A	N/A	*NSL-KDD:* Accuracy: 97.4% FPR: 1.24% *CICIDS2017:* Accuracy: 98.7% FPR: 0.82%	*NSL-KDD:* Accuracy: 96.2% FPR: 1.42% *CICIDS2017:* Accuracy: 96.5% FPR: 1.26%	N/A

defense strategies by building an autonomous IDS that uses learning strategies that are comparable to human learning [222]. Yet it has several limitations such as a need for huge training data and high computational requirements. For example, when solving real-world problems with high dimensions, as the number of inputs increases, so does the number of calculations which increases the overall complexity of DRL algorithms [223]. Training an interactive DRL agent for IDS needs a simulated or real environment [222]. Since training in a real environment is costly, most of the DRL-based intrusion detection research so far employs simulated training environments

which may not provide the same efficiency as real environments. Building more realistic environments that can fully exploit the potential of DRL for effective network intrusion detection needs further investigation.

3.6 Adversarial Attack Realism

Over the past few years, adversarial attack generation has emerged as a highly captivating research area for cybersecurity experts. Several works emphasize enhancing the detection capability of IDS by employing adversarial training, class balancing, and data augmentation techniques [7, 57, 58, 66], it is important to note that most of them are not suitable for real network scenarios where the validity of network data is crucial. Additionally, they have not focused on synthetic attack quality analysis to check for the effectiveness of adversarial attacks.

Some recent cybersecurity research works focus on the investigation of the validity of adversarial cyberattacks. Merzouk *et al.* [205,224] investigate the application of adversarial attacks in real network scenarios. Based on multiple invalidation criteria, a comprehensive analysis of adversarial attacks synthesized using various methods such as the Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), DeepFool, Jacobian-based Saliency Map Attack (JSMA) and Carlini&Wagner's attack (C&W) is provided. The authors identify certain invalidation criteria such as invalid value ranges, invalid binary values, invalid category memberships, and invalid semantic relations to examine the validity of the synthesized adversarial attacks. The results indicate that the majority of attacks generated using such adversarial sample generation techniques are likely unrealistic since they do not follow the necessary network constraints.

Vitorino *et al.* [69] introduces an Adaptative Perturbation Pattern Method (A2PM)

to generate realistic adversarial attacks based on several network domain and class-specific constraints. To keep the functionality of an attack, certain encoded features such as protocol are kept constant while class-specific constraints are modified. The synthesized adversarial attacks are examined in an enterprise network and an IoT network and compared against the corresponding original flows. Multilayer Perceptron (MLP) and Random Forest (RF) models are employed and their performance is compared with and without adversarial training. The detection rate for MLP and RF dropped significantly with adversarial attacks. The augmentation of adversarial synthetic attacks to the training data for MLP and RF improves their performance against such attacks. Although the results show that A2PM produces valid adversarial attacks, the manual selection of features and perturbation of feature values is costly in terms of complexity.

Apruzzese *et al.* [185] provide an elaborate survey for the analysis of state-of-the-art research using adversarial attacks against ML-based IDS. The authors observe that current threat models are invalid for real network scenarios since the attacker has full access and knowledge of target systems. For generating realistic adversarial attacks, similar to real-world attacks, the attacker has minimum or no knowledge of the target systems. The authors highlight three criteria for realistic generation and evaluation of adversarial attacks such as maintaining the nature of the attack, analyzing the interdependency among multiple features, and having legitimate and in-range feature values.

Mozo *et al.* [191] argue that current solutions using GAN to synthesize adversarial data cannot generate better quality attacks due to the convergence issue in GAN training. To handle this issue, their work employs two Wasserstein GANs for synthesizing both normal and attack traffic. Best attack and benign generators are selected by measuring the F1_score of the Random Forest-based IDS on synthetic

data. Here the F1_score obtained by evaluation of the IDS on adversarial data acts as the stopping criteria for GAN training. To measure the similarity between real and synthetic data, two statistical metrics, L1 distance, and Jaccard Coefficient are employed. However, the performance of WGAN dropped when using L1 distance and Jaccard Coefficient indicating that these coefficients are not suitable for adversarial attack quality analysis.

Although some initial work has been done for examining and comparing the quality of synthesized adversarial attacks generated using techniques such as FGSM, BIM, DeepFool, JSMA, and C&W. Nevertheless, the main focus of these research works is to identify adversarial attacks that are challenging to implement for practical scenarios. In contrast, a significant aim of our research is to introduce a framework to synthesize, detect, and analyze the quality of polymorphic adversarial attacks through several validation techniques.

Chapter 4

Problem setup and Assumptions

In this chapter, we outline the fundamental assumptions guiding this research work. We assume that the distribution of normal network data remains relatively constant and consistent across different networks. The problem setup and associated assumptions lay the foundation for exploring the effectiveness of our methodology in countering atypical/polymorphic attacks within network security. Given below, we discuss the threat model and the defense model separately.

4.1 Threat model

The threat model revolves around an adaptive and intelligent attacker aiming to evade detection by dynamically changing the characteristics of network attacks. The attacker leverages AI-based and non-AI techniques to craft polymorphic attacks, challenging the IDS's ability to identify and classify adversarial instances.

4.1.1 Attacker's objective

- *Evade the IDS:* The primary goal of the attacker is to evade detection by the IDS, aiming to disrupt the target. This is achieved by rapidly altering the feature profile and generating atypical/polymorphic attacks.

4.1.2 Attacker's knowledge

- *Test Data Distribution:* The attacker is presumed to know the test data distribution used for training the IDS model.

- *Limited Knowledge of IDS:* The AI-based attacker does not have complete knowledge of the inner workings and decision-making processes of the IDS.

- *Feedback Access:* The attacker has access to feedback from the IDS in the form of a loss function, indicating its effectiveness in synthesizing realistic attacks.

4.1.3 Attacker's capability

- *Computational Capacity:* The attacker is assumed to have the computational capacity to produce atypical/polymorphic attacks.

- *Realistic Attacks:* Starting from a known/original attack, the AI-based attacker can produce adversarial examples that are statistically close to the original attacks and are misclassified by the IDS.

- *Non-AI Attack Tools:* For non-AI attacks, the attacker may employ common attack tools, such as Slowloris DoS, to launch atypical/polymorphic attacks.

- *Source Ambiguity:* When launching non-AI polymorphic attacks, the target is assumed to be unaware of the exact source, as the attacker uses spoofed IDs.

Even if the IDS identifies an attack and blocks a specific IP address, the attacker can relaunch a new attack with a different spoofed IP address.

- *Attack Class Switching:* The polymorphic attacker can not only change the feature profile but can also switch to different classes of attacks, enhancing evasion strategies. For instance, the attacker can initiate the first attack using the Slow Httptest DoS attack tool in Slowloris mode. The subsequent attack using the same tool may be launched in Slow body mode, followed by slow read mode, and so forth.

4.2 Defense model

The defense model, embodied by the IDS, focuses on objectives, knowledge, and capabilities to effectively identify and counteract atypical/polymorphic attacks.

4.2.1 Defense's objective

- *Identify Rapidly-evolving Attacks:* The main objective of the defender (IDS) is to correctly identify and classify the attacks launched by the attacker, particularly those that undergo rapid evolution.

4.2.2 Defense's knowledge

- *Access to Data:* The AI-based IDS has unrestricted access to both training and evaluation data obtained from the benchmark dataset. Moreover, it acquires adversarial data after each adversarial training phase through incremental cycles. Utilizing system logs and information from the target system, the IDS identifies undetected attacks, incorporating this knowledge for subsequent training.

4.2.3 Defense's capability

- *Identify Attacks:* Leveraging the availability of adversarial data for incremental training, the IDS demonstrates the ability to differentiate between realistic synthesized attacks and benign samples in the subsequent attack cycles.

In summary, the threat model outlines potential challenges and goals for the attacker, while the defense model defines the objectives and capabilities of the IDS in countering atypical/polymorphic attacks generated by the adversary in the context of network security.

Chapter 5

Supervised AI Against Atypical/Polymorphic Attacks

The first phase of this thesis presents an assessment of supervised learning-based approaches in detecting atypical/polymorphic cyberattacks. The key goal of this phase is to investigate how the changes in the attack feature profile or an atypical attack influences the performance of the supervised IDS. Previous researchers have deduced that their supervised solutions are effective in the detection of anomalous behavior in networks. However, they have not comprehensively illustrated the effectiveness of their IDS models on attacks with rapidly evolving feature profiles. Given the increased sophistication of cyber attackers today, it is imperative to examine the performance of AI-based IDS for such cases. This will help us understand the classification capability of the supervised IDS on atypical attacks.

Figure 5.1 depicts the working of a generic AI-enabled Network Intrusion Detection System (NIDS). In this scenario, the IDS is assumed to be installed on the network switch, responsible for analyzing incoming or outgoing network traffic to identify potential attacks. An attacker launches a network attack on a target server within the

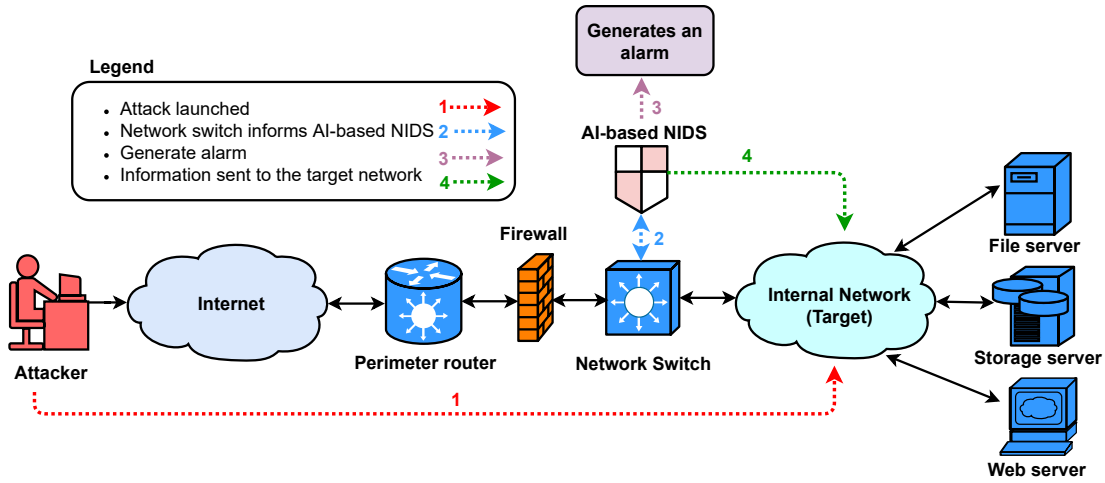


Figure 5.1: Working of an Intrusion Detection System in the network.

internal network. This malicious data is sent to the perimeter router which forwards it to the firewall and ultimately reaches the network switch (if it goes undetected by the firewall). The network switch, equipped with the AI-based IDS capability, assesses the incoming data to determine if it constitutes an attack. Subsequently, it generates an intrusion report and transmits it to the internal network. The network administrator can then take action by blocking this attack and providing guidance for its further mitigation.

5.1 A Generic AI Framework for Supervised IDS

In this work, a generic AI framework for supervised IDS [2] based on the twofold ensemble feature selection technique, and hyperparameter optimization is employed for defense against atypical attacks. The framework consists of four important phases namely, feature preprocessing and selection, training and validation, evaluation and generalization, and hyperparameter optimization (HPO). Figure 5.2 represents a generic framework for a supervised AI-based IDS. This approach can be applied by other researchers to any AI model specific to their problems.

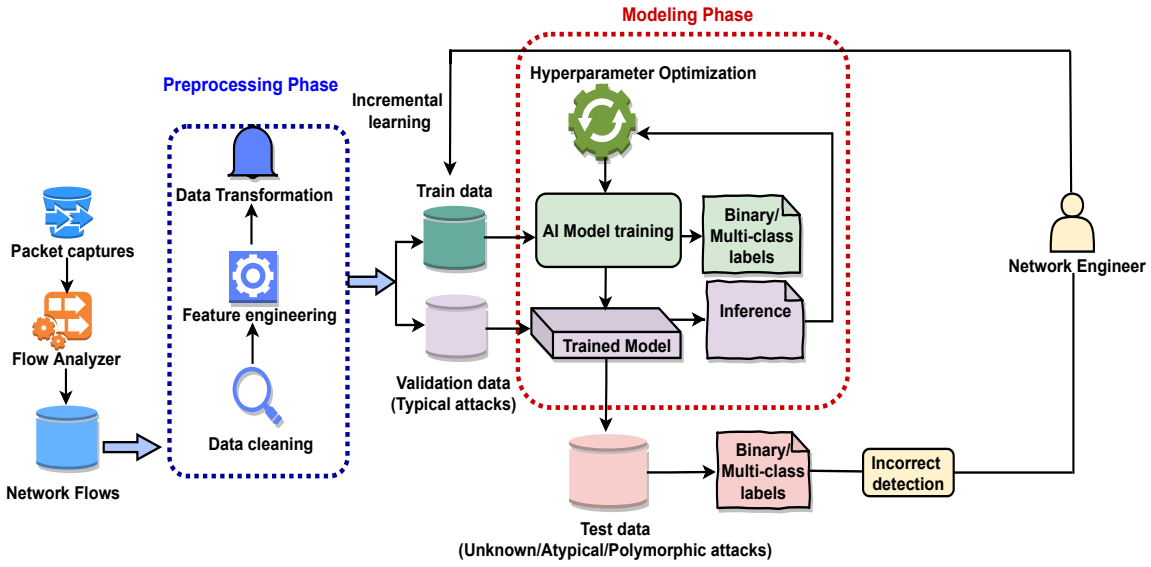


Figure 5.2: A generic supervised AI framework for detecting typical, unknown, atypical, and polymorphic network attacks [2].

5.1.1 Preprocessing and feature selection

As a standard AI practice for intrusion detection, raw network traffic is first converted to traffic flows using a flow analyzer. The synthesized data is then assessed for its quality and cleaned. This phase includes observation and removal of any irrelevant bits, mismatched data types/values, missing data, and outliers. Other major steps for preprocessing include feature extraction/dimension reduction and data transformation.

Offline Feature Selection Technique

In this phase, the clean benchmark data undergoes the feature selection process which consists of two main phases: feature pre-screening and Heterogeneous Feature Selection Ensemble (HFSE). The features in this case represent flow-based information about the entire network such as IP addresses, network ports, protocols, connection

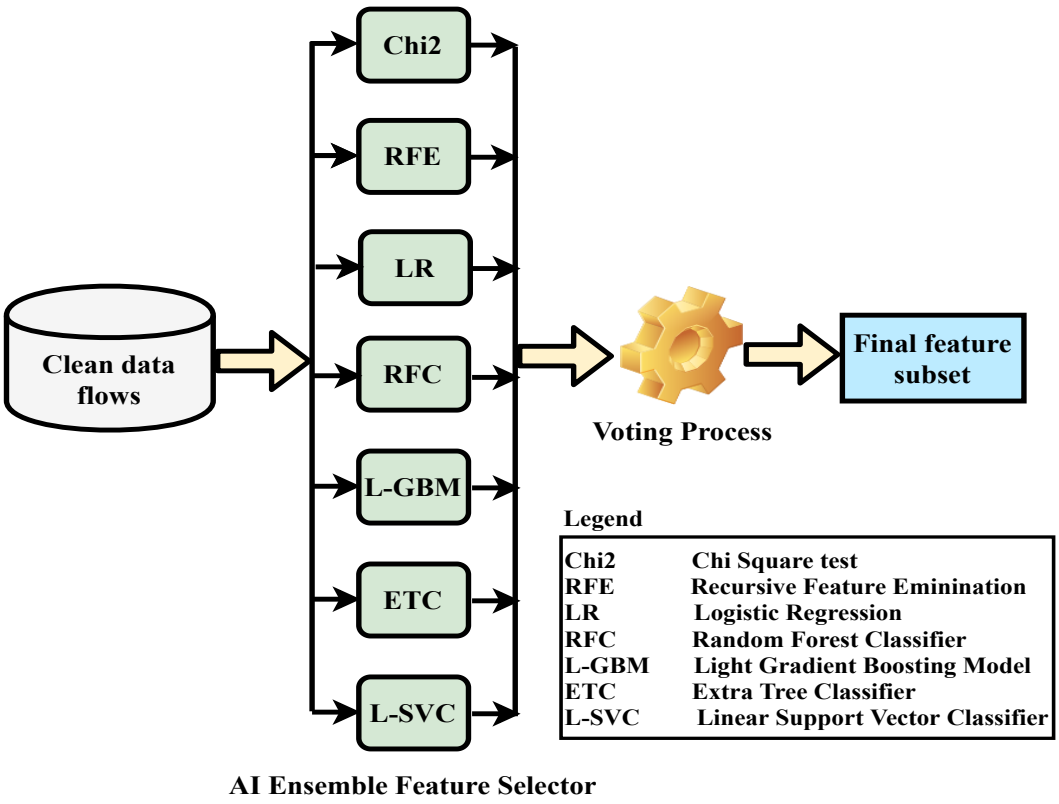


Figure 5.3: Offline Heterogeneous Feature Selection Ensemble (HFSE) [2]

time duration, arrival times, data/packet size, and some other miscellaneous flags. Figure 5.3 represents the feature selection technique applied. A detailed description of the two phases involved in the feature engineering technique is given in the following sections.

Feature Pre-screening

Most AI systems are black boxes providing limited intuition as to why certain predictions were made. Some explainable algorithms like LIME [225] can be used to resolve this issue by explaining the AI predictions based on the selected features, but they do not work efficiently for all types of AI models [226]. When the training data does not fairly reflect important features, the AI algorithm will learn insignificant patterns that

do not aid in improving the detection rates, leading to AI bias [226]. Also, AI-based models may pick up a feature that is insignificant from the point of view of a network attack because such models are trained on currently available public datasets that are often created by standard attack generator tools that have a configured range for various feature values. As such, these AI models may pick up a contributing feature simply because the attack generator was configured to use a fixed feature profile. In such cases, an initial manual feature pre-screening identifies the priority features that reflect the vulnerabilities of the target system.

While we do not know the feature profile of the attack in advance, we do have some information on how an attack could potentially affect the target. Manual pre-screening is performed based on the vulnerabilities of the target. For example, the packet size might not be a determining factor in a SYN DoS attack on a web server, so it may be eliminated in the pre-screening process. However, the same factor could be important in a DDoS attack on a networking device, thus it could be included in the features that the defensive AI considers. In other words, pre-screening focuses on the target vulnerabilities that can be determined offline. While in our process the manual pre-screening is done at the start, it can also be revisited over time when the system states change. However, it remains offline and not part of the dynamic real-time AI response. The refined feature set is then passed through the Heterogeneous Feature Selection Ensemble for further filtering. The main advantage of using this approach is to ensure that the AI selects the most significant features that make the results more accurate, reduce AI bias, and improve performance against atypical attacks.

Heterogeneous Feature Selection Ensemble (HFSE)

Ensemble Feature selection combines the results of multiple models strategically to find the best feature subset. The feature subset selected manually further under-

goes filtering using the Heterogeneous Feature Selection Ensemble in an offline mode. Multiple feature selectors as described above are chosen to be a part of this ensemble as the name implies. The feature selectors in this ensemble vote for the best features using a ranking score. The feature that receives the highest votes is considered the best while the feature with the lowest votes is the least significant. Fig. 5.3 represents our proposed offline feature selection ensemble technique. The main advantage of this approach is to build a hypothesis using multiple models [227] and then combine their results to achieve a better outcome. Using a variety of models for selecting features is advantageous because it controls the variance, and reduces the likelihood of poor feature selection [228].

Data Standardization Process

The selected feature subset is inspected again using a Quantile-Quantile plot (QQ Plot) to check for normality [79]. We apply the z-score normalization on the data to create a Gaussian Distribution with a zero mean and a standard deviation of one [229]. In this case, the feature values are not bound to a specific limit. It is also known as Z-score scaling. The equation for Z-score normalization with y_i as the feature vector, $y^{'}$ as the mean of the feature vector, and σ_i as standard deviation is given in eq.(5.1).

$$Z - score = \frac{y_i - y^{'}}{\sigma_i} \tag{5.1}$$

Data standardization is an important step for most AI algorithms, especially gradient descent-based, such as neural networks, logistic regression, and distance-based algorithms like SVM, KNN, and K-means [230]. These models may not behave as expected [231] if data is not standardized. For gradient descent-based models, feature values affect the step size of the gradient descent [230]. If the features have different

ranges, this might cause variations in the step sizes of each feature. The gradient descent will converge smoothly towards the minima when the features have the same scale [230]. Distance-based algorithms use distances between different observations to classify them into different categories. If some features have different ranges, chances are that they will have a higher impact on the AI results, therefore leading to AI bias. By introducing scaling, all features will contribute equally to generate the results [230]. Moreover, standardization is robust to outliers and new data since it does not have a limited range and can aid in faster convergence of loss functions for some algorithms.

5.1.2 Synthesizing Non-AI Atypical/Polymorphic Attacks

For this research, we assume that the attacker may employ sophisticated publicly available attack tools with the ability to change attack parameters to launch an atypical attack. To simulate this scenario, we have created a virtual network environment. Within this setup, an attacker, represented by a Kali Linux virtual machine, initiates genuine attacks on a target Apache server (deployed on a virtual machine running a Windows operating system). The attacks are conducted using commonly available DoS tools such as Slowloris and Slow Httptest. These slow-rate attacks exploit the HTTP vulnerability and send incomplete requests to the target server to open as many connections as possible. The target's resources and connections are kept engaged while denying access to legitimate users thus leading to a Denial of Service attack [232]. While Slowloris is a slow HTTP header attack, the Slow Httptest attack can be launched in different modes such as slow header, slow body, and slow read [232]. Figure 5.4 depicts a low bandwidth HTTP DoS Attack. Such attacks may be difficult to trace since the attacker may not send any malicious content when sending multiple requests to overwhelm the target server.

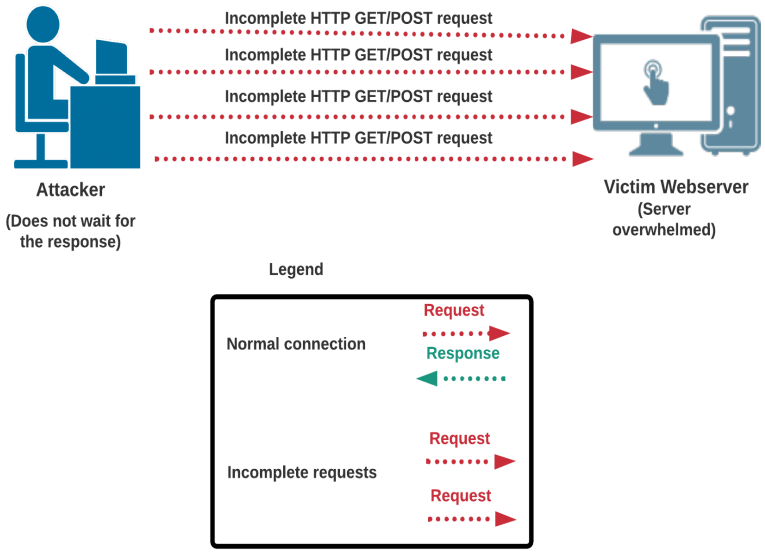


Figure 5.4: Low Bandwidth HTTP DoS Attack

The atypical/polymorphic attacks are synthesized after running Denial of Service (DoS) attack traffic on a target server. All the attacks are separately launched successfully from the attacker to the Apache Server installed on the target machine by sending incomplete GET/POST requests to the target server. The feature profile of the attack is changed by mutating the tool-based parameters from their default values to randomized values to launch a new attack. Packets are captured using the Wireshark tool running in the background on the target machine. The captured data is then analyzed offline and converted into different features using the network flow analyzer *CICflowmeter* [233, 234].

We validate the success of these attacks on the target server by checking its status for the duration of the attack to confirm access disruption. Fig. 5.5 depicts the success of the Slow Httptest DoS attack on the target server. The target's services are successfully disrupted by the attacker after requesting 5000 simultaneous connections. The figure shows service unavailable and target unreachable for the duration of the attack.

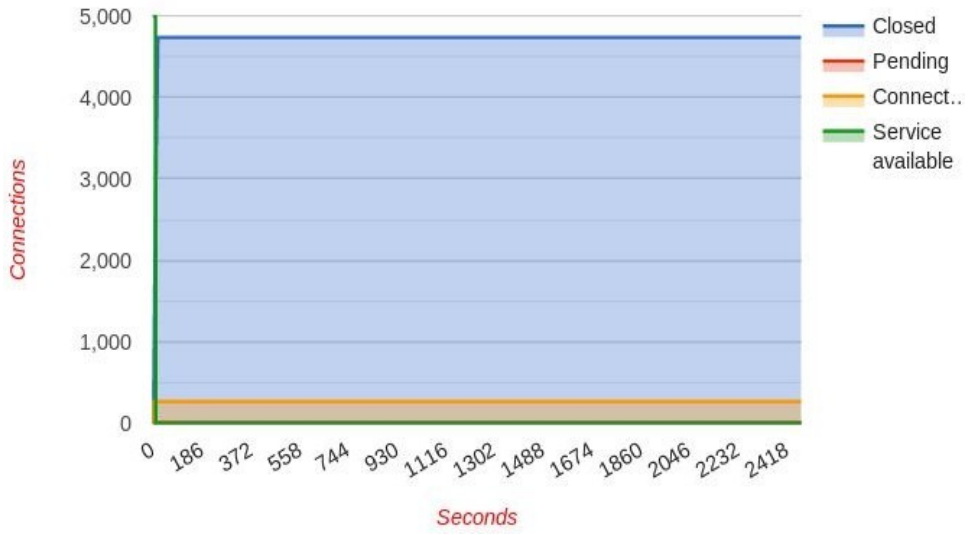


Figure 5.5: A Successful Slow Httptest DoS Attack (Slow Header Mode) on the Target Apache Server.

5.1.3 Training and Validation

The next step for the AI framework in intrusion detection is the training of the AI classifier with the preprocessed benchmark dataset. As a standard practice, the dataset is divided into 3 parts: training, validation, and evaluation. After the model is trained on the training dataset, its performance is observed on the validation data by employing several validation techniques such as simple cross-validation or K-fold cross-validation. In the latter, for each iteration, the data is divided into K-partitions of equal size, one part kept for validation and the rest for training. The validation loss value of each iteration is calculated and then averaged to find the final loss for measuring the model performance.

5.1.4 Evaluation or Generalization

Generalization is an important phase for an AI model in intrusion detection. It essentially signifies how good our classifier is at learning normal observations and

typical attack observations from the benchmark dataset and applying this knowledge to detect atypical attacks that it has never seen before. Mostly, the evaluation of the trained AI classifier is done using the test data subset kept aside before training. This dataset contains normal observations and typical attack observations. To fully capture the classifier's performance or generalization capability, it is essential to evaluate the classifier on a real data subset that is not a part of the original benchmark dataset. This dataset may contain typical and atypical attack observations as well as normal observations. This step is essential to make certain that the observed classifier is not overfitting, provides unbiased results, and will be successful when launched in a real network environment [2].

5.1.5 Hyperparameter Optimization (HPO)

In real-world scenarios, attackers can mutate the attack feature profile quite often thus making it difficult for the AI-based IDS to identify such newly evolved attacks. These systems need continuous or dynamic updating of their models for performance improvement on such newly evolved attacks. Figure. 5.6 depicts the continuous training process with hyperparameter optimization for supervised learning.

We employ HPO for supervised AI models to optimize their training process. This module is designed to generate multiple hyperparameter pipelines and select the best pipeline that results in a robust high-quality AI model for detecting atypical attacks. This technique generates models that are already optimized and ready to be deployed for the classification of newly evolved attacks. The main goal is to optimize the system by improving the AI model building process until the performance of such models on newly synthesized atypical attacks is improved. The performance is measured based on the True Positive Rate (TPR) for attack flows and the True Negative Rate (TNR) for normal flows. The optimization process is done when the models do not generalize

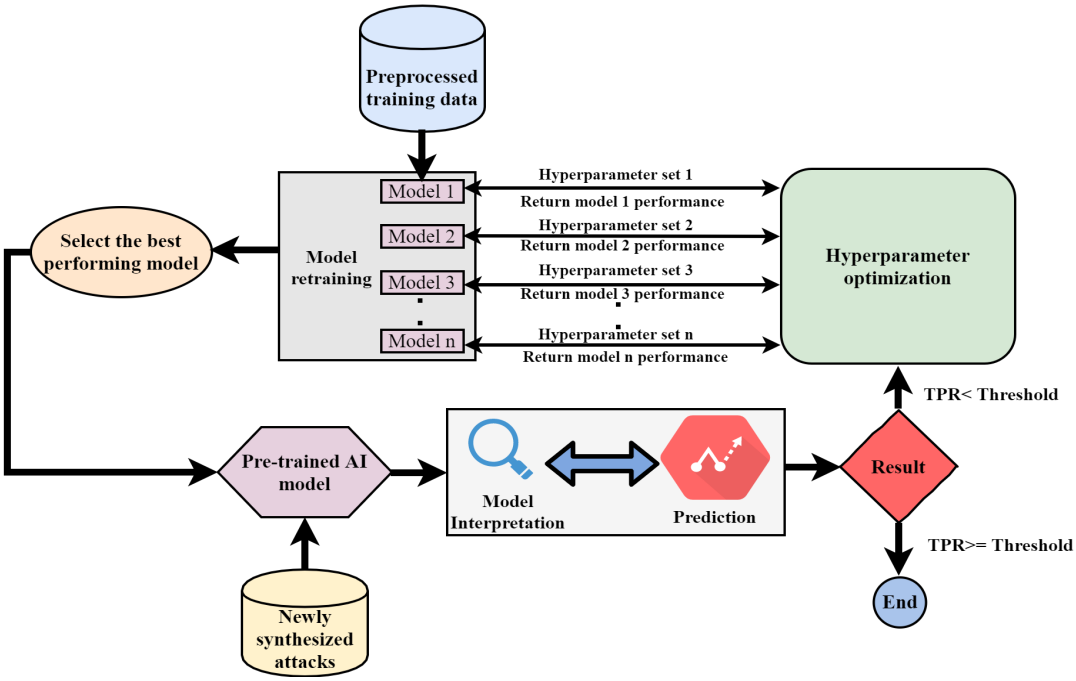


Figure 5.6: Continuous Training with Hyperparameter Optimization

well on atypical attacks with TPR less than a predefined threshold value. The model is then retrained again offline until the TPR no longer improves.

The workflow representing the employed supervised approach to deal with atypical attacks is shown in Figure. 5.7. The generic workflow can be applied to any supervised AI model for training, generalization, and optimization. Algorithm 5.1 represents a generic overview of the methodology for synthesizing and identifying atypical attacks. The algorithm begins by initializing the TPR (Recall) threshold value to T. This is followed by our offline feature selection and training steps. The trained AI model is then applied to synthesized atypical attack flows for testing and performance evaluation. The performance metrics used are accuracy, precision, recall, and F1-score. For hyperparameter optimization, the recall value is compared with the predefined threshold (T) set initially. If the value is less than T, the model continues retraining repeatedly until the optimal performance is achieved.

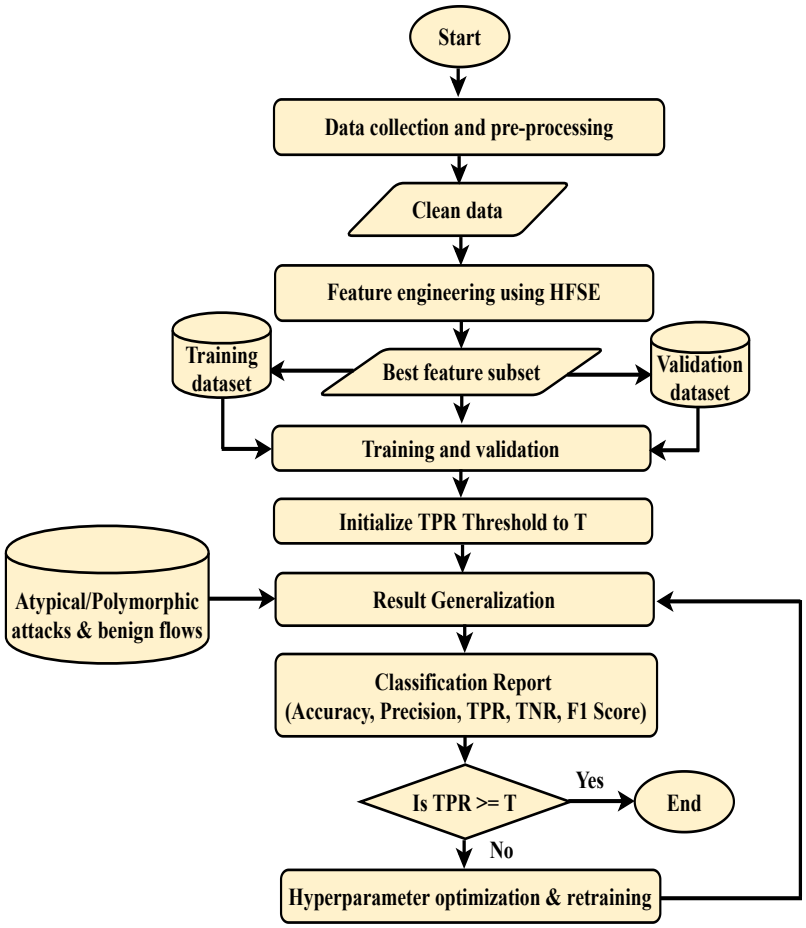


Figure 5.7: Workflow for Atypical Attack Detection using Supervised Learning [2]

5.2 Experiments and Evaluation of Supervised AI

This section explains in detail the training and validation of supervised ML/DL classifiers using CICIDS2017 data and their performance measurement. The machine used for this research to conduct the experiments consists of the following configurations: Intel Xenon CPU E5-2630 v3@ 2.40GHz, 480GB SSD, 64-bit Windows 10 Pro, and NVIDIA Quadro K2200. The Hyperparameter optimization step for various AI models was implemented on a server installed with 4 GPUs (GeForce GTX 1080 Ti with a compute capability of 6.1).

Algorithm 5.1: Atypical Attack Generation & Identification using supervised AI [2]

Input:
IDS input- Original training data d_{tr}
Test data d_{ts} with typical attacks d_{ta} and benign flows d_{be}
Number of iterations for atypical attack generation - n

Output:
Atypical Attacks d_{aa}
Trained IDS

initialize TPR threshold as T;

/ Attack Generation */*
for $i = 1$ to n **do**
  Attacker A synthesizes atypical attacks d_{aa} by randomly changing attack parameters
end for

while *true* **do**

  / Training the IDS */*
  Train and validate IDS using d_{tr} and d_{ts}
  Generalize IDS results using atypical attacks d_{aa} and benign flows d_{be}

  / Examine TPR on d_{aa} */*
  if $TPR < T$ **then**
    Hyperparameter Optimization();
  else
    Break;
  end if
end while

5.2.1 CICIDS2017 Dataset Details

CICIDS2017 benchmark dataset [233, 234] is used only for training and validation of the AI models against typical attacks and benign observations. It consists of several attack classes and benign flows. More details are given in subsection 7.1.1. Table 5.1 shows the best 20 features selected using the proposed feature selection technique based on their respective rank in the voting score.

5.2.2 Atypical Data Details

We synthesized 8 atypical attacks, 4 each for two low-rate DoS attack classes, Slowloris and Slow Httptest DoS. These attacks have similar features as that of training data,

Table 5.1: Best feature subset selected using HFSE Technique [2]

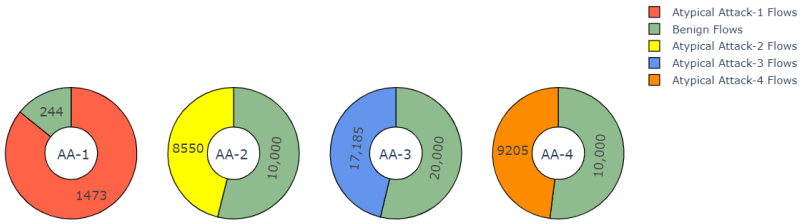
S.No.	Feature name	Chi2	RFE	LR	RFC	L-GBM	ETC	L-SVC	votes
1	pkt len var	✓	✓	✓	✓	✗	✓	✓	6
2	pkt len std	✓	✓	✓	✓	✓	✓	✗	6
3	max pkt len	✓	✓	✓	✓	✗	✓	✓	6
4	fwd pkts/s	✓	✓	✓	✗	✓	✓	✓	6
5	fwd IAT max	✓	✓	✓	✗	✓	✓	✓	6
6	flow IAT max	✓	✓	✓	✓	✓	✓	✗	6
7	init win bytes fwd	✓	✗	✗	✓	✓	✓	✓	5
8	fwd pkt len mean	✓	✓	✓	✓	✗	✓	✗	5
9	fwd IAT std	✓	✓	✗	✓	✗	✓	✓	5
10	fwd IAT mean	✓	✓	✓	✗	✓	✗	✓	5
11	flow IAT std	✓	✓	✓	✗	✓	✗	✓	5
12	pkt len mean	✓	✗	✗	✓	✗	✓	✓	4
13	min pkt len	✓	✓	✓	✗	✗	✗	✓	4
14	flow IAT mean	✓	✓	✗	✗	✓	✗	✓	4
15	flow duration	✓	✗	✓	✗	✓	✓	✗	4
16	avg pkt size	✓	✗	✓	✓	✗	✓	✗	4
17	fwd pkt len std	✗	✓	✓	✗	✗	✗	✓	3
18	fwd pkt len min	✓	✓	✗	✗	✗	✗	✓	3
19	fwd pkt len max	✗	✓	✗	✓	✗	✓	✗	3
20	fwd IAT tot	✓	✗	✗	✗	✓	✓	✗	3

Table 5.2: Features for Synthesizing Slow Httptest DoS Attacks

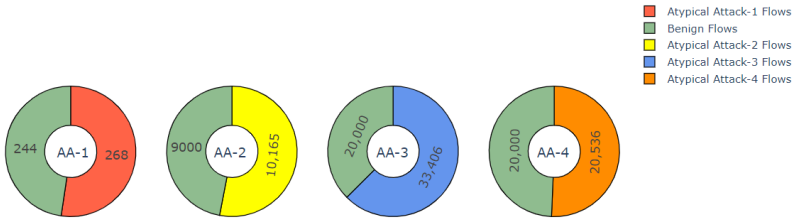
Feature	Description
c	number of target connections
i	followup data interval in seconds
r	number of connections per second
l	length of attack in seconds
t	request verb (GET/POST)
x	maximum length of followup data per tick
p	timeout to wait for http response in seconds
s	size of content length header in bytes
w	start range of advertised window in bytes
y	end range of advertised window in bytes
n	interval between read operations in recv buffer in seconds
z	bytes to slow read from recv buffer with single read call
k	number of times to repeat the same request in the connection

but have a different range of values of features and are synthesized in different network settings as compared to training data.

Slow Httptest attack is launched in 3 different modes namely Slow Header (H), Slow Body (B), and Slow Read (X) to create different feature profiles for the attack. The features used for synthesizing Slow Httptest atypical attacks are explained in detail in Table 5.2. We mutate these tool-based features from their default values to random values to generate new feature profiles. The synthesized atypical attacks with different feature profiles are represented in Table 5.3. The CICflowmeter (flow



(a) Slow Httptest Atypical Data



(b) Slowloris Atypical Data

Figure 5.8: Atypical Attack and Benign Flows. AA is used to depict feature profiles for Atypical Attack (1-4) both for Slow Httptest and Slowloris Attack classes.

analyzer) is used to generate flows from raw attack packet captures. The number of attack flows generated depends on the time duration of the attack. If the attack time duration is higher, a greater number of attack flows will be generated.

As shown in Fig. 5.8a, the number of attack flows for Slow Httptest atypical attack-1, attack-2, attack-3, and attack-4 are 1473, 8550, 17185 and 9205 respectively. We would like to indicate that in our case, the number of atypical attack flows is expected to be small since our intention is precisely to examine how a supervised AI-based IDS can deal with mutated attacks that are not present in the training data. We have selected benign flows randomly from CICIDS2017 test (hold-out) data to match the number of attack flows for analyzing AI models using different performance metrics. The total number of random benign flows for Slow Httptest DoS atypical attack-1, 2, 3, and 4 are 244, 10000, 20000, and 10000 respectively.

For Slowloris DoS, we use target port number (p), number of sockets (s), and random user agents (ua) features to generate atypical attacks. The random user agents

Table 5.3: Different Feature Profiles for Slow Httptest DoS Attack. AA represents an atypical attack.

AA	Mode	Duration	# Flows	c	i	r	l	t	x	p	s	w	y	n	z	k
AA1	H	240 sec	1473	1000	10	300	240	GET	24	3	-	-	-	-	-	-
AA2	H	2475 sec	8550	5000	25	500	10800	GET	24	10	-	-	-	-	-	-
AA3	B	7200 sec	17185	5000	120	500	7200	POST	10	10	4096	-	-	-	-	-
AA4	X	1800 sec	9205	10000	10	450	1800	GET	32	5	-	512	1024	5	32	3

Table 5.4: Different Feature Profiles for Slowloris DoS Attack

Atypical Attack	Duration	# Flows	p	s	ua
AA1	900 sec	268	80	265	No
AA2	3600 sec	10165	80	500	No
AA3	10800 sec	33406	80	1000	No
AA4	7200 sec	20536	80	5000	Yes

feature is used to randomize user agents for each request. The details of atypical slowloris attacks with different feature profiles are given in Table 5.4. As shown in Fig. 5.8b, Slowloris atypical attack-1 consists of 268 flows while atypical attack-2 consists of 10165 attack flows, attack-3 and attack-4 consist of 33406 flows and 20536 flows respectively. For Slowloris DoS atypical attack-1, 2, 3, and 4, the number of randomly selected benign flows are 244, 9000, 20000, and 20000 respectively.

5.2.3 Training and Validation Phases

After the feature selection phase, we train and validate different ML/DL classifiers such as Deep Neural Network (DNN), Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN), Bernoulli Naïve Bayes (BNB), Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Linear Support Vector Classifier (L-SVC), Linear Discriminant Analysis (LDA), Decision Tree Classifier (DTC) and Random Forest Classifier (RFC) on the CICIDS2017 data. All ML classifiers are trained with their default hyperparameter configurations on 75% train data.

To train a DNN model, 20 input dimensions are used (equal to the number of selected features). The model has 3 Dense layers with Rectified Linear Unit (ReLU)

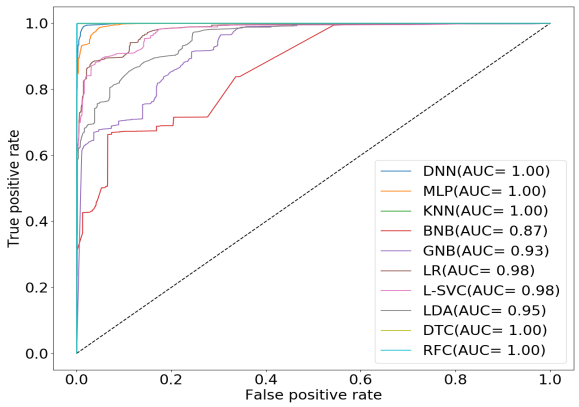


Figure 5.9: ROC Comparison of AI models on CICIDS2017 Test Data (Typical Attack and Benign Flows)

activation functions, each has 60 neurons. Each Dense layer is followed by Dropout with a value of 0.2 to avoid overfitting. The last Fully Connected (FC) layer along with Sigmoid activation provides output probabilities for attack or benign traffic. The model is trained for 200 epochs, but early stopping is applied to reduce overfitting. The batch size is set to 1024 and the learning rate is 0.0001.

Fig. 5.9 shows a comparison between various ML/DL models based on their AUC values on CICIDS2017 test data which consists of typical/known attacks. This hold-out data consists of 262143 flows (both attack and benign) and constitutes 25% of the total CICIDS2017 data flows. AUC measures the ability of a classifier to distinguish between classes by calculating the area under the Receiver Operating Characteristic (ROC) graph. The ROC graph represents the plot between TPR and FPR at various threshold values. AUC values lie in the range from 0 to 1. The higher the AUC value, the better the model performance. Fig. 5.9 shows that the best AUC is achieved by models such as DNN, MLP, KNN, DTC, and RFC. Other models such as LR and L-SVC also achieve AUC values as high as 0.98. Models such as BNB, GNB, and LDA also show similar improvements.

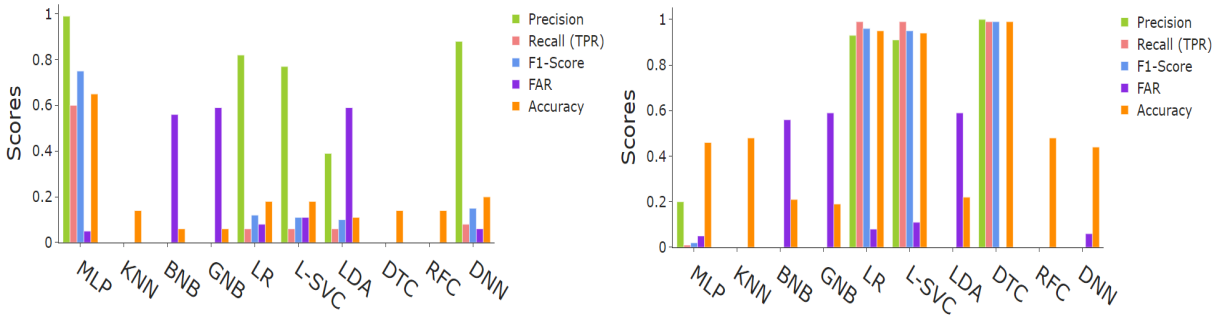
Table 5.5: Performance of AI Models on Atypical Attack-1 & Benign Flows

Atypical Attack-1 (AA1)			
Model	**TNR**	**TPR (Slow Httptest)**	**TPR (Slowloris)**
DNN	93.44%	8.35%	0%
MLP	95.08%	59.88%	1.12%
KNN	100%	0%	0%
BNB	43.85%	0%	0%
GNB	40.98%	0%	0%
LR	91.80%	6.31%	98.88%
L-SVC	89.34%	5.97%	98.88%
LDA	45.49%	5.70%	0%
DTC	100%	0%	98.88%
RFC	100%	0%	98.88%

5.2.4 Model Generalization and Evaluation

The pre-trained ML/DL models are evaluated against synthesized atypical attacks to improve model generalization. We conduct this evaluation before HPO to analyze which AI model can identify these attacks with higher detection rates. We compare the performances of these models using one atypical attack (atypical attack-1) from both the attack classes namely, Slow Httptest and Slowloris. Other synthesized atypical attacks are used to evaluate multiple AI models after the hyperparameter optimization phase. The TNR for benign flows and TPR for attack flows are separately measured as shown in Table 5.5. Although most models provide high TNR on CICIDS2017 test data, they generally have poor performance on the atypical attack-1 from both Slow Httptest and Slowloris attack classes. LR, L-SVC, and DTC can detect only Slowloris atypical attack-1 with 98.88% TPR and benign data with TNR of 91.80%, 89.34%, and 100% respectively. All other models perform poorly on both attacks.

Fig. 5.10 provides a comparative analysis of different AI models using multiple performance metrics including precision, recall, F1 score, FAR, and accuracy. The figure shows that LR, L-SVC, and DTC have a better performance on Slowloris atypical attack-1 while MLP performs better on Slow Httptest atypical attack-1 only.



(a) Slow Httptest Atypical Attack-1

(b) Slowloris Atypical Attack-1

Figure 5.10: Evaluation of AI Models on Atypical Attack and Benign Flows

This signifies that the studied AI models are unable to perform well in identifying both classes of atypical attacks. The reason for this poor performance on atypical attacks is due to the overfitting problem of these models. Therefore, we subject the AI classifiers to further optimization of hyperparameters to improve their generalization on atypical attacks.

5.2.5 Hyperparameter Optimization

Since the TPR of the AI models employed for evaluation on atypical attack-1 is less than the predefined threshold value (in our case, 80%), these models need HPO to further improve their performance. For simplicity, we selected the models that could identify at least one of the synthetic atypical attack classes during evaluation to undergo further optimization. These models are L-SVC, DNN, and DTC. Other AI models discussed in this thesis can also be optimized using this technique but they are not examined further. We also point out that our approach is generic and can be applied to any AI-specific model.

For the hyperparameter optimization phase, AI models are trained with the CI-CIDS2017 train data using multiple hyperparameter (HP) settings. Then, their performances are evaluated against atypical attack-1 data (both Slowloris and Slow

Httptest DoS) to select the best set of hyperparameters. Further evaluation of these optimized AI models is done using multiple atypical attacks as discussed in this section.

HPO and Generalization for DNN

We apply Manual Search Hyperparameter Optimization (MS-HPO) on the DNN model to improve the TPR and TNR for attack and benign flows, respectively. The input dimensions for model architecture are 20 since our feature selection technique uses 20 features. The model has 3 Dense layers with Rectified Linear Unit (ReLU) activation functions. The first two layers have 1024 neurons while the third dense layer has 512 neurons. Each Dense layer is followed by Dropout with a value of 0.3 to avoid overfitting. The last Fully Connected (FC) layer along with Sigmoid activation provides output probabilities of synthesized test data being 'attack' or 'benign'. These parameters are kept constant during the MS-HPO process. We have only utilized batch size (bs), number of epochs, and learning rate (lr) for HPO purposes. We conducted the MS-HPO for the DNN model with different batch sizes, number of epochs, and learning rates for a total of 10 iterations out of which only the best 5 are discussed further in this paper.

Table 5.6 shows the results of hyperparameter tuning on the DNN model. We observe that the last two hyperparameter configurations provide better results than others. We select the last configuration with a learning rate of 0.0005, batch size of 16, and 25 epochs. Our goal is to improve the TPR of attack flows as well as TNR for benign flows, thereby reducing both FNR and FPR. After employing our proposed HPO technique for the DNN model, the TNR for Slow Httptest atypical attack-1 improves by 51.26% and by 100% for Slowloris atypical attack-1 as compared to the DNN model discussed in Table 5.5.

Table 5.6: Effect of MS-HPO on DNN model

Atypical Attack-1			
HP	**TNR**	**TPR (Slow Httptest)**	**TPR (Slowloris)**
lr-0.001 bs-1 Epochs-12	76.23%	0.00%	0.00%
lr-0.001 bs-6 Epochs-15	97.95%	6.31%	0.00%
lr-0.0008 bs-8 Epochs-10	97.95%	58.04%	1.11%
lr-0.001 bs-8 Epochs-15	**32.79%**	**69.93%**	**100%**
lr-0.0005 bs-16 Epochs-25	**93.85%**	**59.61%**	**100%**

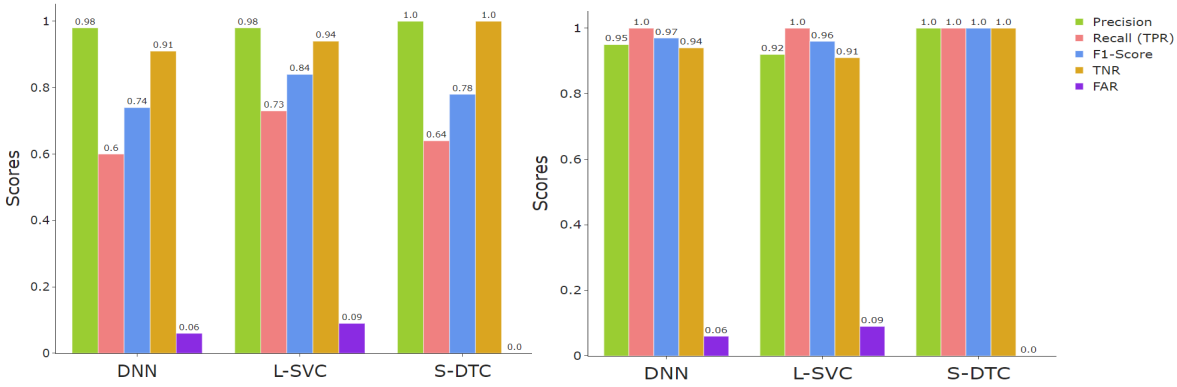
Table 5.7: Hyperparameter Optimization for Linear Classifiers

Model	**HP**	**Description**	**Values**
L-SVC	C	Regularization	[0.8,1.0,1.5,2.0]
	tol	Tolerance for stopping	[0.001,0.01,0.1,1]

HPO and Generalization for Linear-SVC

Since L-SVC is the best performing model among linear classifiers after the feature engineering phase, we select it for further performance improvement using GS-HPO. The CICIDS2017 data is divided into two datasets, 70% for training and 30% for validation. The model is evaluated on Slow Httptest Atypical Attack-1 and Slowloris Atypical Attack-1. Table 5.7 discusses the hyperparameters (HP) used in GS-HPO for the L-SVC model.

The L-SVC model is retrained based on the best hyperparameters selected by Grid Search Hyperparameter Optimization (GS-HPO) which are *C=1.5* and *tolerance=1.* Here C represents the regularization parameter while *tolerance* represents the tolerance for stopping criteria [235]. It is observed from Fig. 5.11 that after the HPO phase, the performance of L-SVC for benign flows improves by 1.64% whereas, for Slow Httptest Atypical Attack-1, TPR improves by 67.01% and for Slowloris Atypical



(a) Slow Httptest Atypical Attack-1 (b) Slowloris Atypical Attack-1

Figure 5.11: Evaluation of AI models after HPO on Atypical Attack-1 and Benign Flows

Attack-1, it improves by only 1.12% as compared to the L-SVC results depicted in Table 5.5.

HPO and Generalization for Tree-Based Models

We employ the Tree-Based Optimization Tool (TPOT) [236], which is based on Evolutionary Search Hyperparameter Optimization (ES-HPO) explained in the background section of this paper, to find the right set of hyperparameters for building the AI model. This optimization process took 48 hours with a population size of 50 and 5 generations to find the right hyperparameter set for Stacked Decision Tree Classifier (S-DTC). The *population size* in this case is a positive integer that depicts the number of individuals in a population. *Generations* is also a positive integer representing the number of iterations to run the pipeline optimization process [236]. The resultant S-DTC combines the power of two DTCs stacked together in the ensemble. The first classifier has a maximum depth of 10, minimum leaf samples of 4, and minimum split samples of 20. The second classifier has a maximum depth of 10, minimum leaf samples of 10, and minimum split samples of 16.

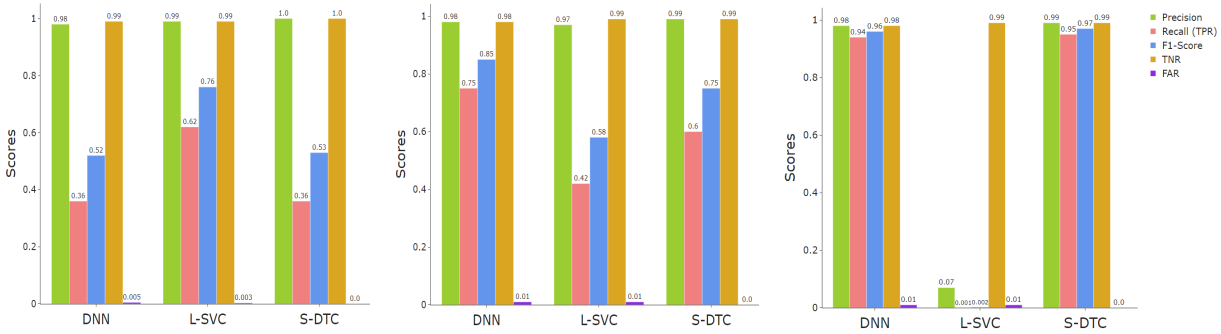
Fig. 5.11 represents the performance evaluation of DNN, L-SVC, and S-DTC

Table 5.8: A Comparison Between Our Approach and Current Research for Typical, Atypical Attacks and Benign Flows. Here, Typical Attack Represents the Hold-out Attack from CICIDS2017. ShttpAA1 Represents Slow Httptest Atypical Attack-1 and SlowAA1 Represents Slowloris Atypical Attack-1.

Model	**TNR** *Benign*	**TPR** *Typical*	**TNR** *Benign*	**TPR** *ShttpAA1*	**TPR** *SlowAA1*
L-SVC [our approach]	98.69%	75.70%	90.98%	**72.98%**	**100%**
DNN [our approach]	96.25%	96.07%	93.85%	59.67%	**100%**
S-DTC [our approach]	**99.92%**	**99.90%**	**100%**	64.02%	**100%**
L-SVC [237]	98.66%	77.93%	92.62%	5.70%	98.88%
DNN [120]	98.57%	80.75%	94.67%	57.50%	1.11%
CNN [238]	99.76%	97.11%	100%	0%	0%
LSTM [239]	98.64%	98.57%	97.13%	18.67%	0%

models after HPO in terms of multiple performance metrics such as precision, recall, F1 score, and FAR. The results show an improvement in the performance metrics for these models as compared to Fig. 5.10. For example, the recall value for L-SVC on Slow Httptest is improved from 5.97% (before HPO) to 72.98% (after HPO). For DNN, the recall value has improved from 8.35% (before HPO) to 60% (after HPO). As seen in the figure, the AI models perform well when evaluated on Slowloris atypical attack-1 with all the performance metric values equal to or above 92%. It is observed that L-SVC has a better performance in terms of TPR, and F1 Score as compared to DNN, and S-DTC. However, the S-DTC gives the lowest FAR of almost 0% as compared to other AI models.

Table 5.8 shows a comparison of L-SVC, DNN, and S-DTC models trained using our approach with current research for both typical and atypical data based on TPR and TNR values. We train the models presented in [237], [120], [238], and [239] with their default hyperparameters and evaluate their performance using typical attacks, atypical attack-1, and benign flows. Stable results of the AI models trained using our approach on both typical as well as atypical attacks compared to other state-of-the-art models indicate the first-rate performance of our approach.



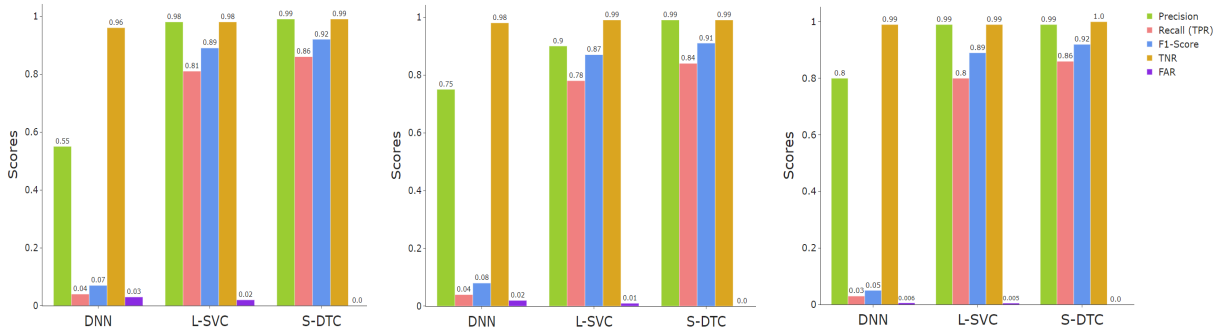
(a) Atypical Attack-2 (b) Atypical Attack-3 (c) Atypical Attack-4

Figure 5.12: Comparison of AI Models on Slow Httptest Atypical Attack and Benign Flows after HPO

5.2.6 Evaluation and Comparison of AI Models on Atypical Attacks

This section provides an evaluation of the performance of AI models namely DNN, L-SVC, and S-DTC against atypical attacks. The atypical attack and benign flows (2 to 4) shown in Fig. 5.8b and Fig. 5.8a are used for further comparison of the AI models' detection capability.

The performance comparison of DNN, L-SVC, and S-DTC on different atypical attacks belonging to the Slow Httptest DoS attack class is shown in Fig. 5.12. All 3 AI models achieve high precision, high TNR, and very low FAR scores indicating their superior performance on benign flows. Yet their performance (Recall) on atypical attack flows is lower and lies in the range from 36% to 95%. For atypical attack-2, L-SVC gives the best performance with 62% recall. The DNN recall value for atypical attack-3 is 75% which is the highest among all the models. For atypical attack-4, DNN and S-DTC have high recall values of 94% and 95% respectively whereas L-SVC was not successful in identifying this attack. The Precision and F1 score values for all these models also vary according to their performances in identifying attack



(a) Atypical Attack-2 (b) Atypical Attack-3 (c) Atypical Attack-4

Figure 5.13: Comparison of AI Models on Slowloris Atypical Attack and Benign Flows after HPO

and benign flows. We observe that none of the models can identify all 3 attacks with high recall values. DNN and S-DTC achieve higher recall rates as compared to L-SVC which performs poorly on atypical attack-3 and 4.

Fig. 5.13 compares the performances of DNN, L-SVC, and S-DTC on Slowloris atypical attacks-2, 3, and 4. The results depict a similar performance on all three attacks since all the attacks are launched in a slow header mode. These attacks have similar features to the typical attacks in the training dataset but have a mutated feature profile. In comparison, Slow Httptest DoS atypical attacks in Fig. 5.12, depict different results. One of the reasons is that these atypical attacks are launched in three different modes namely, Slow header, Slow body, and Slow read. Therefore, they belong to three different categories of slow-rate DoS Attacks launched using the same attack tool.

For Slowloris DoS Atypical attacks in Fig. 5.13, DNN, L-SVC, and S-DTC achieve high TNR and a very low FAR for all the cases which implies that benign flows can be accurately identified. From the results on atypical attack-2, 3, and 4, DNN is the worst performing model for the identification of these attacks with recall values of 4%, 4%, and 3% respectively. Whereas L-SVC and S-DTC achieve higher recall

(a) Slow Httptest Atypical Data

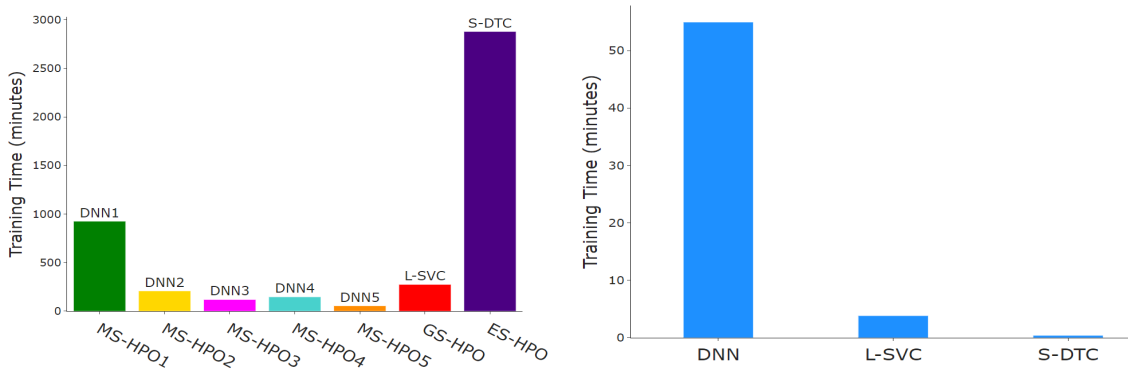
(b) Slowloris Atypical Data

Figure 5.14: Accuracy Comparison of AI models after HPO on Typical and Atypical Data. Note that Typical Attacks (TA) Represent the Attacks in the Hold-out CI-CIDS2017 Data and AA represents an Atypical Attack.

values on all three attacks indicating that for these cases, they provide better atypical attack identification as compared to DNN. For L-SVC, the recall values for atypical attack-2, 3, and 4 are 81%, 78%, and 80% respectively. Results on atypical attack-2, 3, and 4 indicate that for the Slowloris attack class, S-DTC is the best performing model with recall values of 86%, 84%, and 86% respectively.

We provide an analysis of the overall accuracy of the AI models on typical attacks, multiple atypical attacks from Slow Httptest DoS and Slowloris DoS attack classes as well as benign flows in Fig. 5.14a and Fig. 5.14b respectively. All the models, DNN, L-SVC, and S-DTC achieve higher accuracy on typical attack data as compared to atypical attack data. For Slow Httptest DoS, the overall accuracy is lower as compared to Slowloris DoS attacks. The accuracy for L-SVC on Slow Httptest Atypical attack-4 is the lowest around 50% approximately. For DNN and S-DTC, the accuracy is equal to or greater than 65%. For Slowloris DoS, L-SVC, and S-DTC in general have a higher accuracy on atypical attack data as compared to DNN which is the worst-performing model on most of the atypical attacks.

Although we observe some enhancement of model performances against atypical



(a) Total Run Time for HPO Techniques (b) Total Train Time for AI Models after HPO

Figure 5.15: Time Complexity Analysis for Hyperparameter Optimization (HPO) Techniques and Selected AI Classifiers after HPO. DNN(1-5) Represent Different Hyperparameter Configurations for DNN Model (Discussed in Table 5.6). Here, MS-HPO Represents Manual Search HPO, GS-HPO Represents Grid Search HPO, and ES-HPO Represents Evolutionary Search HPO.

attacks after the HPO phase, further optimization and retraining are needed to meet the pre-defined threshold values for recall. Overall results indicate that S-DTC performs relatively better than all the models in comparison except on Slow Httptest atypical attack-2 where its recall value is only 36%. The other two models in comparison, DNN, and L-SVC also have lower recall values (36% and 62% respectively) on this attack. We believe that the higher misclassification rate for atypical attacks may be due to the under-representation of this attack feature profile in the training data. We plan to investigate this poor performance issue, re-optimization of AI models, and retraining phases in our future work using an incremental learning approach.

We follow the approach in [240] to provide a complexity analysis for the hyperparameter techniques employed in this research as well as for training the AI models in Fig. 5.15. As shown in Fig. 5.15a, we compare the total run time required to train a DNN model with different sets of hyperparameters for identifying atypical attacks (MS-HPO) with GS-HPO for L-SVC, and ES-HPO for S-DTC. After analysis, the results indicate that ES-HPO for the S-DTC takes the longest time (2880 minutes) to

identify the hyperparameters for atypical attack detection. We train the AI models DNN, L-SVC, and S-DTC with the set of hyperparameters selected during the HPO process for each model. Our results highlight that after the HPO phase, the training complexity for DNN, L-SVC, and S-DTC is reduced as indicated in Fig. 5.15b. S-DTC takes the least time to train among all the models whereas DNN takes the longest time of 54.98 minutes owing to the complex deep learning model structure.

Our main motivation for undertaking this research stems from the lack of an extensive analysis of the effectiveness of supervised IDS models against atypical attacks. To enhance the current knowledge of dynamically changing attacks, we introduce the defensive AI engine to build and analyze IDS models against such attacks. Some important factors to consider while building an AI model for IDS are the size of available training data, the number of features, and the training time required. The first important step for our methodology is selecting the most relevant features to minimize training loss and to make AI interpretations easier.

From our investigation through this research, we deduce that the selection of optimal hyperparameters is a very important step in building enhanced IDS models that can face evolving attack strategies. However, this process can be very intensive, time-consuming, and complex. Although we see improvements in the performances of our selected AI models over multiple atypical attacks, they need more optimization of hyperparameters and further retraining to meet predefined TPR thresholds.

We employed only two classes of slow-rate DoS attacks, Slowloris and Slow Httptest to synthesize atypical attacks for IDS evaluation. Such attacks may be difficult to trace since the attacker may not send any malicious content when sending multiple requests to overwhelm the target server.

Extensive improvements in the results may be expected especially with the recent advances in adversarial learning and other semi-supervised approaches. These

techniques use adaptive algorithms to generate network behavior that has never been seen before. This newly synthesized network data can be employed to improve the performance of the IDS against dynamically changing attacks. Henceforth, for the next part of our research, we explored adversarial semi-supervised methods to defend against atypical/polymorphic attacks.

Chapter 6

Polymorphic Attack Generation and Detection using CVAE-AN

Today's networked systems face significant security challenges with the increased sophistication of rapidly changing (polymorphic) attacks. Especially with the emergence of adversarial attacks, the attackers can evade detection by the AI-enabled intrusion detection system (IDS). Small alterations are added to the network attack traffic to create adversarial attacks that can impact the decisions of the AI-based IDS. The goal of such sophisticated attacks is to manipulate an attack instance in such a way that it is deceptively classified as benign. In this thesis, we focus on improving the performance of IDS against such polymorphic attacks (both adversarial and non-AI) using our proposed CVAE-AN model.

A generic framework representing our methodology is depicted in Fig. 6.1. This framework serves as the architectural backbone, providing structure and organization to the concepts and solutions we have introduced through this research. This framework consists of five important phases namely, data collection and preprocessing, feature engineering and SHAP analysis, adversarial attack generation and detection

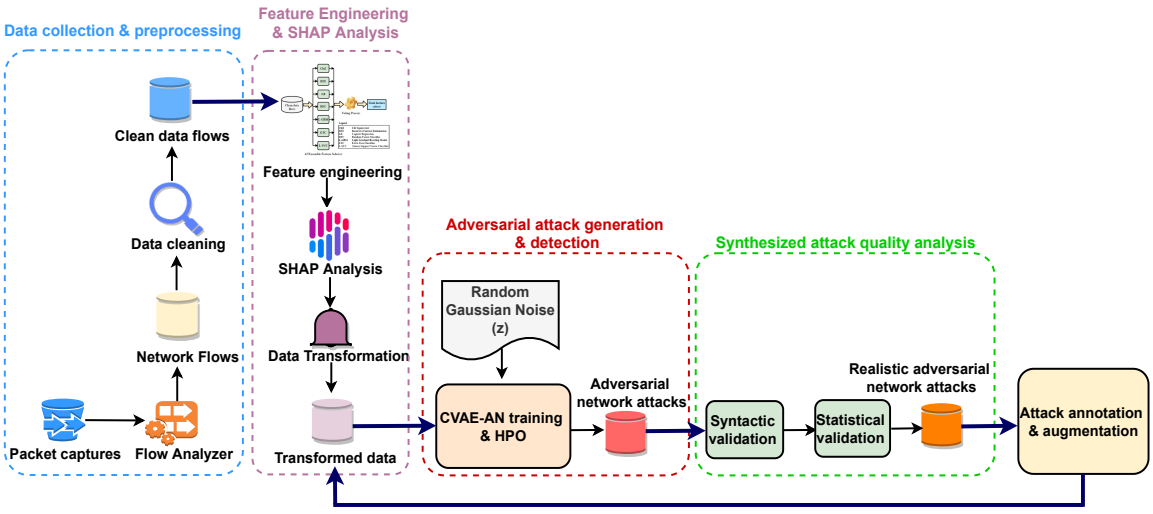


Figure 6.1: A generic framework for realistic polymorphic attack generation and detection.

using CVAE-AN, synthesized attack quality analysis, and data annotation and augmentation.

6.1 Network data collection and preprocessing

Network packets are captured and passed through a flow analyzer to extract data flows. These network flows undergo preprocessing which includes cleaning and feature extraction following a standard ML practice. We adopt the data preprocessing phase discussed previously in subsection 5.1.1.

6.2 Feature engineering and SHAP analysis

6.2.1 Feature selection

In this work, the best training feature subset is identified by employing the Heterogeneous Feature Selection Ensemble (HFSE) [2] as discussed in subsection 5.1.1.

HFSE is a voting ensemble of AI-based feature selector methods such as Chi-square, Recursive Feature Elimination, Logistic Regression, Linear Support Vector Classifier, Random Forest, Extra Tree Classifier, and Light Gradient Boosting Machine. The votes from each method for a feature under observation are counted to select the best feature. Features with the largest number of votes are chosen to be a part of the final feature subset.

6.2.2 Preserving Functional Attack Features (SHAP Analysis)

To preserve the functionality of the attack, we apply SHapley Additive exPlanations (SHAP) [241] Explainable AI (XAI) method to CICIDS2017 and CICIoT2023 binary class data. SHAP selects the best features that determine the functional attack behavior. These functional features have a favorable effect when identifying each attack. SHAP employs the concepts of coalitional game theory and Shapley values to explain each prediction. In this case, every feature is considered a 'player' in the game, and Shapley values give an average estimate of each feature towards a prediction. SHAP is explained in eq.(6.1).

$$m(k') = \phi_0 + \sum_{i=1}^{N} \phi_i k'_i \tag{6.1}$$

Here, $m(k')$ is the prediction or output for the instance k', ϕ_0 is the base value representing the average or expected output, $\phi_i \in R$ is the contribution for feature i in the prediction or Shapley value for feature i, k'_i represents the value of the ith feature for the instance k', $k' \in \{0,1\}^N$ represents the feature vector, and N represents the maximum coalition size or total number of features.

This process is advantageous since it gives us valuable insight into each feature's

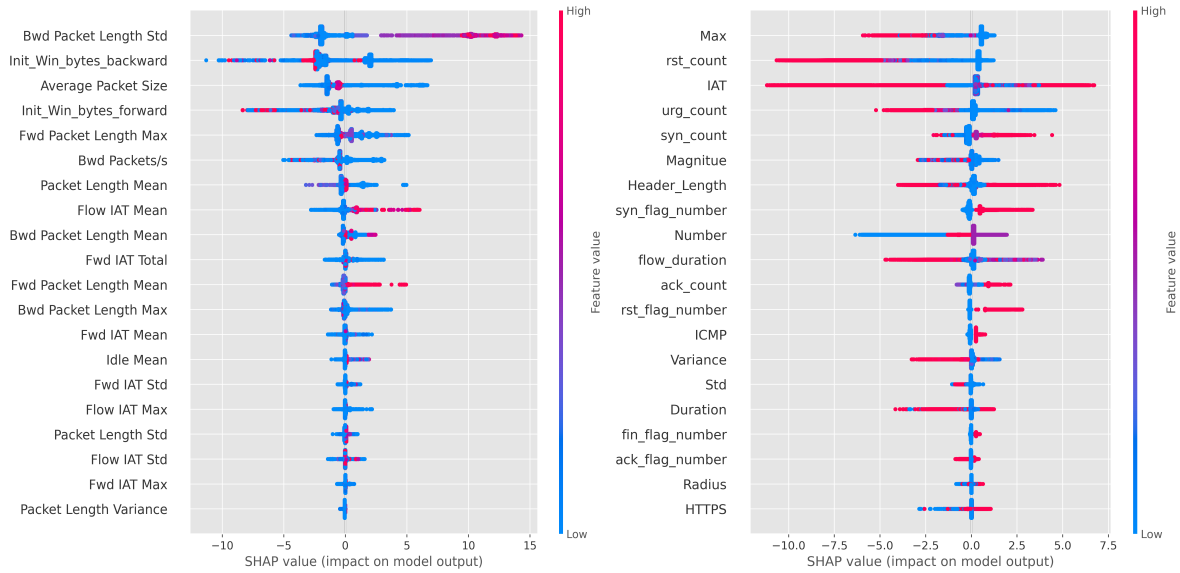
(a) Slow Httptest DoS Attack

(b) Slowloris DoS Attack

Figure 6.2: Explaining an accurately identified attack within the Slow Httptest DoS and Slowloris DoS category by utilizing SHAP

contribution towards attack identification for a data flow under observation.

Fig. 6.2 explains the detection results for Slow Httptest DoS and Slowloris DoS attack classes using SHAP. In Fig. 6.2a for the Slow Httptest DoS attack class, the base value of 0.34 represents the average of all predictions made by the IDS model on training data. $f(x)$ represents the output value or prediction of the model which equals 0.64. The features in pink influence positively and drag the prediction closer to 1 whereas the features in green influence negatively and drag the prediction towards 0. In Fig. 6.2a, the major positive contribution is from features 8, 18, 7, and 10 and the major negative contribution is from features 1, 12, 9, and 6. Similar behavior is seen in Fig. 6.2b when detecting a Slowloris DoS attack. From both the figures, a greater number of features exert influence on the prediction pushing it towards the right, as opposed to those that are pushing the prediction towards the left. This results in prediction values of 0.64 and 0.99 for Slow Httptest DoS and Slowloris DoS respectively. This proximity to 1 signifies that the observation has been correctly classified as an attack. This pattern is also observed in other attack classes, hence those instances are not detailed in this discussion. When this analysis is extended to the entire dataset, the major features that positively impact an attack are identified



(a) CICIDS2017 DDoS/DoS dataset (b) CICIoT2023 dataset

Figure 6.3: SHAP summary plot for CICIDS2017 and CICIoT2023 datasets respectively.

as functional or attack features. These features are further employed to preserve the functional nature of an attack when synthesizing an adversarial attack.

Fig. 6.3 illustrates SHAP summary plots for the CICIDS2017 and CICIoT2023 datasets, respectively. Each point on the graph corresponds to a flow (row) from the input dataset. The color of each point represents the impact of a feature value on attack detection output, with red indicating high values and blue indicating low values. For instance, in Fig. 6.3a, the feature *'Bwd Packet Length Std'* exhibits a high impact on the model's ability to detect attacks as indicated by the red color. Similarly, in Fig. 6.3b, the feature *'syn_flag_number'* demonstrates a significant impact on the model's ability to identify attacks, indicated by the red color. This pattern is observed for other features as well, showcasing their impact on attack detection.

6.3 Adversarial attack generation and detection

6.3.1 Model design and adversarial training

The preprocessed network attack flows along with random Gaussian noise are fed as inputs to a trained generative DL model. For this purpose, we introduce Conditional Variational Autoencoder Adversarial Network (CVAE-AN) [3, 70], an enhancement of a Semi-supervised Generative Adversarial Network (SGAN) model [28–30]. In a simple SGAN model, the standard generator suffers from the mode collapse problem and therefore cannot synthesize diverse data samples [200]. While WGAN is less prone to mode collapse and provides a more stable training process than a standard GAN, it cannot synthesize multiple variations of a given input class. For this objective, we employ a CVAE-based generator instead of a standard generator in the SGAN model. A CVAE is better suited since it can synthesize data with specific conditions and has full control over the generated results [171]. For our research, the main goal is to synthesize multiple variations of atypical attacks, each with a different feature profile, a CVAE-based attack generator is the most appropriate model for this task.

We built the CVAE-AN model for training the CVAE-based attack generator and the discriminator in an adversarial environment. The discriminator model has an unsupervised and a supervised component. The unsupervised discriminator discovers hidden patterns in the input unlabeled data and identifies real from synthesized data similar to a standard GAN discriminator. The supervised discriminator acts as a binary classifier (IDS) to identify attacks from benign observations based on the patterns learned by the unsupervised discriminator. Fig. 6.4 depicts our novel unified CVAE-AN model for polymorphic attack generation and detection.

The general objective of the attacker is to synthesize dynamically changing atypi-

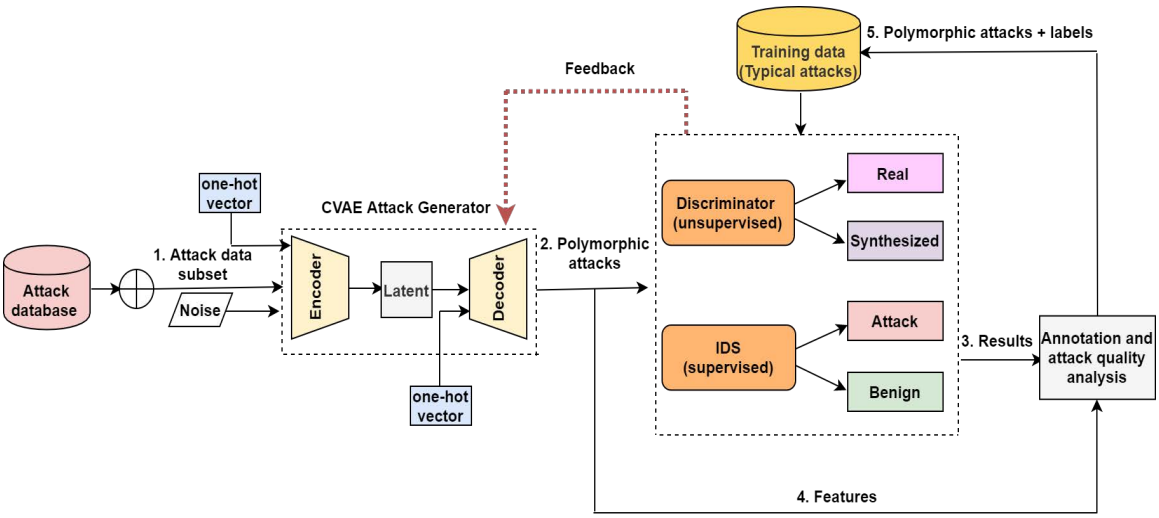


Figure 6.4: Polymorphic attack generation and detection using CVAE-AN.

cal attacks/polymorphic attacks and elude discovery by the IDS. On the other hand, the general objective of the IDS is to continuously improve the attack detection rate against polymorphic attacks.

The initial stage of adversarial learning involves training the CVAE attack generator using input attack data and Gaussian noise tailored for each attack class. The Gaussian noise is added as a source of randomness and helps to create diversity in the input data. The attacker learns input attack patterns and synthesizes new variations of an attack from the underlying data distribution and categorical one-hot vector information (class label). The discriminators are initially trained using benign and typical attack samples from the benchmark datasets. The supervised discriminator (IDS) identifies typical/polymorphic attacks from benign samples. The unsupervised discriminator recognizes generated polymorphic attacks from real input data samples. Annotated polymorphic attack samples are then added to the input data in the subsequent training cycle to enhance the overall attack detection rate for the IDS.

Optimization procedure for the attacker and the IDS

The detailed optimization procedure for CVAE-AN involves training the model components to minimize the defined objective functions. Here are the main steps:

Forward pass: For training the CVAE generator G, Gaussian noise with specific mean and standard deviation values and input attack data X along with the condition variable (one-hot vector) y is fed into the encoder of the CVAE which produces latent representation z. This latent information is used by the decoder to generate the reconstructed sample X'. The unsupervised discriminator D_{unsup} and the IDS (supervised discriminator) D_{IDS} are trained with the benchmark dataset (consisting of real attacks and real benign data). The adversarial attack samples synthesized by the generator X' are also passed through D_{unsup} and D_{IDS} for discrimination.

Loss computation: In this phase, two losses are computed for the CVAE-AN model consisting of adversarial loss and CVAE loss. The discriminators provide the adversarial loss indicating how well they distinguish between real and generated samples for D_{unsup} and between attack and benign samples for the D_{IDS}. CVAE loss is a combination of reconstruction loss between X and X' and KL divergence between the learned variational distribution from the latent and a chosen prior distribution (multivariate Gaussian).

The discriminator D_{unsup} maximizes the probability of assigning high scores to real samples $log\ (D_{unsup}(x))$ and low scores to generated samples $log(1 - D_{unsup}(G(z)))$. The adversarial loss for D_{unsup} is denoted in eq.(6.2).

$$L_{adv,D_{unsup}} = -\mathbb{E}_{x\sim p(x)}[log\ (D_{unsup}(x))] - \mathbb{E}_{z\sim p(z)}[log(1 - D_{unsup}(G(z)))] \qquad (6.2)$$

The D_{IDS} maximizes the probability of identifying attacks from benign samples.

The adversarial loss for D_{IDS} is denoted in eq.(6.3).

$$L_{adv,D_{IDS}} = -\mathbb{E}_{x\sim p(x)}[log\ (D_{IDS}(x))] - \mathbb{E}_{z\sim p(z)}[log(1 - D_{IDS}(G(z)))] \tag{6.3}$$

The generator G aims to minimize the combined adversarial loss for both D_{unsup} and D_{IDS}. The total adversarial loss for G is denoted in eq.(6.4).

$$L_{adv,G} = -\mathbb{E}_{z\sim p(z)}[log(D_{unsup}(G(z)))] - \mathbb{E}_{z\sim p(z)}[log(D_{IDS}(G(z)))] \tag{6.4}$$

In addition to the adversarial loss, the CVAE generator is trained using a combination of reconstruction loss L_{recon} and conditional prior L_{KL} represented in eq.(6.5).

$$L_{cvae} = L_{recon} + L_{KL} \tag{6.5}$$

The overall loss for the CVAE-AN model is represented in eq.(6.6).

$$L_{total} = (L_{adv,D_{unsup}} + L_{adv,D_{IDS}} + L_{adv,G} + L_{cvae}) \tag{6.6}$$

Backward pass: During backward pass or backpropagation, the generator or CVAE gradients are computed for CVAE loss and adversarial loss with respect to the generator's parameters. They guide the updates to the generator to enhance its ability to synthesize realistic and diverse adversarial attack samples. Gradients are computed for adversarial losses from D_{unsup} and D_{IDS} which guide updates to the discriminator D_{unsup} to enhance its ability to identify real samples from synthesized samples and to IDS D_{IDS} in classifying attacks from benign.

Parameter updates: The gradients obtained from backward pass are used to update the parameters of the generator and the discriminator using an optimization

algorithm such as Adam. This involves adjusting weights and biases in the direction that minimizes the combined adversarial and CVAE losses. This process is repeated for multiple data batches until convergence.

Information sharing between the generator, unsupervised discriminator, and the IDS

Typically the generator G, unsupervised discriminator D_{unsup}, and the IDS D_{IDS} operate independently during training. G does not have complete knowledge of inner workings and decision making of D_{unsup} and D_{IDS} and vice versa. They adapt iteratively based on the feedback loop. The key information shared during training is related to generated samples and their quality. D_{unsup} and D_{IDS} share feedback with G in the form of a loss function indicating how well the generator is performing in synthesizing realistic network attacks and in evading detection by the IDS. G uses this feedback to adapt its parameters, improving the quality of generated samples making them look like realistic attacks and maximizing the probability of evading detection by the IDS. On the other hand, the discriminator D_{unsup} adjusts its parameters to identify real from synthesized and the D_{IDS} adjusts its parameters to identify attack from benign. D_{unsup} and D_{IDS} serve as critical components in the adversarial training process providing feedback to guide the improvement of $G's$ performance and vice versa.

6.3.2 Hyperparameter Optimization and retraining

The hyperparameters for the CVAE-AN model are chosen randomly by conducting multiple trial runs and experiments until appropriate values are identified. For each run, we evaluate the Evasion Success Rate (ESR) of the synthesized polymorphic attack on the IDS (supervised discriminator). We aim to select the set of hyperpa-

rameters that result in the highest ESR since we want the polymorphic attacker to evade detection by the IDS. Once the best set of hyperparameters is determined, we implement those settings to conduct further training of the CVAE-AN model.

6.3.3 Synthesizing Dynamically Changing Atypical Attacks/ Polymorphic Attacks

The techniques given below are applied to generate dynamically changing atypical attacks/polymorphic attacks:

Adversarial Atypical/ Polymorphic Attacks

Dynamic atypical attacks are synthesized by the polymorphic CVAE attack generator by adding random Gaussian noise with distinct values for mean and standard deviation to the training attack data for each cycle. When the IDS detects the attack, the polymorphic attacker modifies the feature profile again in the next cycle. During each attack cycle, the values for mean and standard deviation for Gaussian noise are altered by smaller amounts, for example, unit increments to synthesize newer noise values while training the attacker which generates another polymorphic attack. This process of polymorphic attack chain continues until the IDS can detect any previously unseen attacks initiated by the attacker or until the attacker has exhausted all the features used to launch a new attack.

Algorithm 6.1 provides a generic outline for generation and detection of an adversarial polymorphic attack using CVAE-AN. The Big O complexity of this model depends upon several factors such as the size of the dataset, the architecture of the generator and the discriminator, and the number of training iterations. For the generator G, if $O(g)$ is the complexity of a single forward pass, then the overall complexity

Algorithm 6.1: Adversarial Polymorphic Attack Generation & Identification based on CVAE-AN [3, 70]

Input:
Noise N with a mean of μ and standard deviation of σ
Generator input - Noise N plus train attack data X_{attk}
Discriminator/IDS input - original training data X_{real} with typical attacks X_{ta} and benign flows X_b
Number of training iterations - *n_steps*

Output:
Trained Generator G, Trained Discriminator D, Trained IDS C

for $i = 0$ to *n_steps* **do**
  $\mu = i$;
  $\sigma = i + 1$;

  /* *Training the Generator G* */

  G synthesizes adversarial polymorphic attacks X_{pa}: $G(N(\mu, \sigma) + X_{attk})$
  Adjust the gradients of G through back propagation

  /* *Training the Discriminator D* */

  D identifies X_{real}, X_{pa}
  C identifies typical attacks X_{ta}, polymorphic attacks X_{pa} and benign flows X_b
  Adjust the gradients of D, C through backpropagation
end for

for the generator training step is $O(n*g)$. Similarly, $O(d)$ is the complexity of a single forward pass for the Discriminator (consisting of D and C). The overall complexity for the discriminator training step is $O(n*d)$. The total complexity of Algorithm 6.1 in Big O notation is $O(n*(g+d))$, where n is the number of training iterations.

Non-AI Synthesized Polymorphic Attacks

For synthesizing non-AI polymorphic attacks, we follow the methodology given in chapter 5 section 5.1.2. Within a virtual network, we employ publicly acquired attack tools to conduct a series of diverse attacks on the target server sequentially. During this process, we introduce random modifications to the parameters of attack tools aiming to synthesize attacks with different feature profiles. These non-AI polymorphic attacks are employed offline to assess the detection ability of the trained IDS on

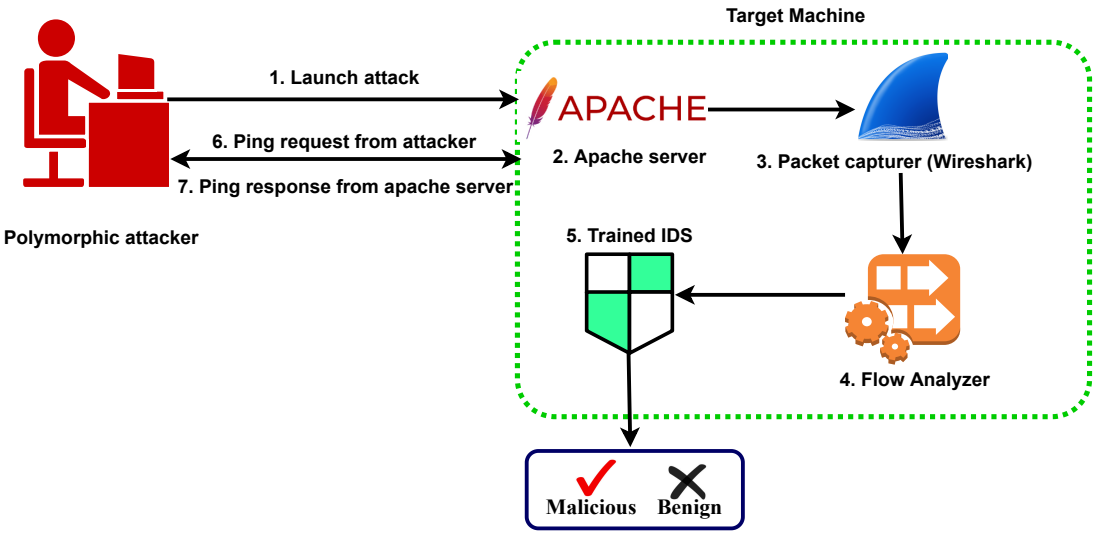


Figure 6.5: Polymorphic attack generation and detection in a virtual network.

realistic network attacks. Figure. 6.5 represents polymorphic attack generation and detection in a virtual network.

6.3.4 Evaluation and testing

The evaluation phase involves validation of our IDS model's performance for each polymorphic attack and retrain cycle using several applied performance metrics such as Balanced Accuracy (BA), IDS proficiency (pID), and Overall Error Rate (OER). We employ the Evasion Success Rate (ESR) metric to measure the performance of the CVAE polymorphic attacker for each cycle. The results are analyzed to understand how well the model is performing.

6.4 Polymorphic Adversarial Attack Quality Analysis

To fully investigate the quality of polymorphic adversarial attacks generated by our system, we first employ syntactic validation of the data. This step involves examining the consistency of the generated attacks with network constraints. Subsequently, we apply several statistical validation techniques, including hypothesis testing, statistical distance-based analysis, and correlation analysis to compare real attack data and adversarial polymorphic attacks. The combination of these statistical metrics provides a comprehensive analysis of the quality of generated attacks in different aspects, including shape, location, overlap, divergence, and correlation. These metrics collectively offer insights into how well the adversarial synthesized attacks align with the characteristics of the original attacks. This aids in assessing effectiveness of the CVAE-AN attack generator in capturing the desired properties of attack data.

The following provides a detailed overview of the syntactic and statistical validation techniques employed for the analysis of the quality of synthesized attacks.

6.4.1 Syntactic validation of adversarial attacks

The quality of adversarial network attacks is studied using several syntactic validation criteria such as range coverage for feature values, the validity of binary values, and valid category membership [205]. Our adversarial synthesized datasets discussed in subsection 6.3.3 are first validated using the following syntactic constraints:

Range coverage for feature values

To maintain the validity of an adversarial attack, the attributes of network data should have an acceptable range of values that follow network constraints. If these values do not lie within the range, the synthesized adversarial data is considered inconsistent. Therefore, when generating adversarial data, it is important to follow the interval range for each feature similar to realistic network attacks. For example, when comparing the range of values for the feature *Average Packet Size* in the synthesized attack data to the original data, it is observed that in our case this value lies within similar ranges for both datasets. Similarly, other synthesized features are also examined for range coverage and compared to the original data.

Validity of binary feature values

The validity of binary feature values is also an important constraint that needs to be maintained in adversarial attacks. Binary features for a network dataset are features that can only have two values, commonly represented as 0 and 1. 0 represents a certain category or non-existence of a specific trait while 1 represents the presence of the trait. The validity of binary values for features such as flags, for example, *Fwd PSH Flags* and other flags in the synthesized dataset is also assessed to make sure that non-binary values are not assigned to binary values.

Validity of category membership

Furthermore, the categorical features that are one-hot encoded (one instance has a value set to 1, and the rest of the instances have a value set to 0) are examined to check that non-zero values should not be assigned to multiple categories at a time for a given instance. For example, the protocol cannot be set to 1 for both TCP and

UDP at the same time.

6.4.2 Statistical validation of adversarial attacks

Kolmogorov-Smirnov hypothesis test

Inspired by research in the image processing field [242], we employ Kolmogorov-Smirnov hypothesis testing (KS test) to verify if the adversarially synthesized atypical/polymorphic attacks and original network attacks belong to the same probability distribution.

A two-sample Kolmogorov-Smirnov test [243] is a nonparametric statistical test that compares the similarity of distributions for a sample from two different datasets. It measures the maximum difference between the cumulative distribution functions of the two distributions. Two hypotheses are considered for this test. The null hypothesis H_0 states that the two samples from the two datasets belong to the same distribution. The alternate hypothesis H_1 states that the two samples belong to different distributions. First, the KS statistic, which measures the distance between the two empirical distributions for all the values of x, is calculated. Then, the corresponding critical value (p-value) for the calculated distance is determined. The p-value indicates the strength of evidence for accepting or rejecting the null hypothesis H_0 based on the observed data [244]. When the p-value is small, it indicates strong proof against the null hypothesis. The p-value is compared with the statistical significance level α which is normally set to 0.05. If the p-value is equal to or larger than the significance level, then the null hypothesis H_0 is accepted and if it is lower than the significance level, then the alternative hypothesis is accepted. The KS-test statistic D_{ks} is explained in eq.(6.7).

$$D_{ks} = \max_{x}(|F_1(x) - F_2(x)|) \tag{6.7}$$

Here, $F_1(x)$ and $F_2(x)$ are the cumulative distribution functions (CDFs) for the two data samples respectively.

Hellinger Distance

Hellinger distance measures the distance between the two probability distributions and its value lies in the range of 0 to 1 [245]. It access the similarity and overlap between original and synthesized datasets. If the distance between the two probability distributions is closer to 1, then the observed distributions are dissimilar whereas if the distance is closer to 0, the distributions resemble each other very closely.

The Hellinger distance D_h is defined in eq.(6.8).

$$D_h = \frac{1}{\sqrt{2}} ||\sqrt{P1} - \sqrt{P2}||_2 \tag{6.8}$$

Here, $P1$ and $P2$ are the probability distributions for the two data samples respectively.

Kullback-Leibler Divergence (KLD)

Kullback-Leibler Divergence (KLD) measures the score of how one probability distribution diverges from another probability distribution [246]. This metric measures the information lost when one probability distribution is used to approximate another. It is also referred to as relative entropy. Although this metric is a distance based metric, it is not symmetric. It calculates the closeness of the two distributions to one another. The value for KLD ranges from 0 to positive infinity. A value of 0 indicates that the two distributions are identical, while a larger value indicates greater dissimilarity.

Given two probability distributions X and Y, the KL Divergence can be calculated as $D_{KL}(X||Y)$ defined in eq.(6.9).

$$D_{KL}(X||Y) = \sum_i X(i)\ \log\left(\frac{X(i)}{Y(i)}\right) \tag{6.9}$$

$X(i)$ is the probability of the i-th event according to distribution X, $Y(i)$ is the probability of the i-th event according to distribution Y. The sum is taken over all possible events in the sample space. The value of KL Divergence $D_{KL}(X||Y) \geq 0$ and is not symmetric which means $D_{KL}(X||Y) \neq D_{KL}(Y||X)$.

Jensen-Shannon Divergence (JSD)

Jensen-Shannon Divergence (JSD) measures the average dissimilarity between the two probability distributions [247]. This metric is a more smoothed/ normalized version of KLD since it is symmetric. The value of JSD ranges from 0 to 1. A value of 0 indicates that the two distributions are similar while a value of 1 indicates that the two distributions are dissimilar.

The JS Divergence for two probability distributions X and Y is measured using $JSD(X||Y)$. JSD is calculated by computing KLD for both X and Y relative to M and then taking their average. This is defined in eq.(6.10).

$$JSD(X||Y) = \frac{1}{2}(D_{KL}(X||M) + D_{KL}(Y||M)) \tag{6.10}$$

Where M represents the midpoint distribution which is calculated as the average of X and Y : $M = \frac{1}{2}(X + Y)$. $D_{KL}(X||M)$ is the KL Divergence between X and the average distribution M. $D_{KL}(Y||M)$ is the KL Divergence between Y and M. Since JSD employs normalized KLD, the divergence of X from Y is the same as Y from X. This is denoted as: $JSD(X||Y) == JSD(Y||X)$.

Correlation Analysis

Pearson's correlation-based similarity metric [248] is used to measure the pairwise feature correlation between two data distributions and analyze their semantic resemblance. The range of this metric lies between 0 to 1. A similarity score of 1 indicates that the pairwise correlations between two data distributions are the same and vice versa for a score of 0. The similarity between the two correlations is calculated using the eq.(6.11).

$$Similarity = 1 - \frac{|A_{p,q} - R_{p,q}|}{2} \tag{6.11}$$

Here, $A_{p,q}$ and $R_{p,q}$ represent the correlation value for the first and second data distribution respectively for a pair of features p and q. The Pearson's correlation value [82] for each data distribution is represented by eq. (6.12).

$$corr = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{6.12}$$

In this context, *corr* is the correlation coefficient, which lies between -1 and 1 and measures the strength of linear correlation between two variables. x_i represents all the values of a feature x in a sample, $\bar{x}$ represents the mean of all values of feature x, y_i represents all the values of a feature y in a sample and $\bar{y}$ represents the mean of all values of feature y.

6.5 Attack annotation and augmentation

After analyzing synthesized adversarial attack quality, the attacks that follow all the syntactic data constraints and are closer in distance to real network attacks are selected for further annotation and are incrementally augmented to input training

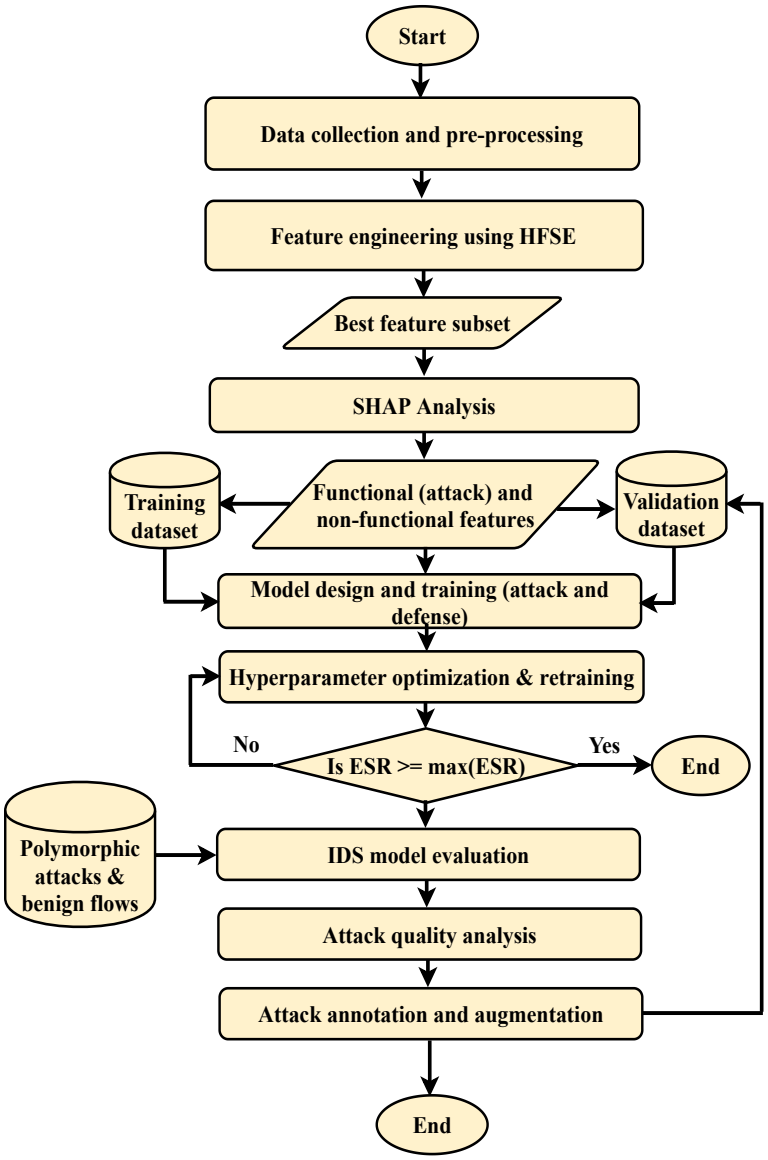


Figure 6.6: Workflow for Polymorphic Attack Generation and Detection

data for class balancing and improving the performance of our IDS against adversarial attacks.

The overall workflow for polymorphic attack generation and detection using our methodology is given in Fig. 6.6.

Chapter 7

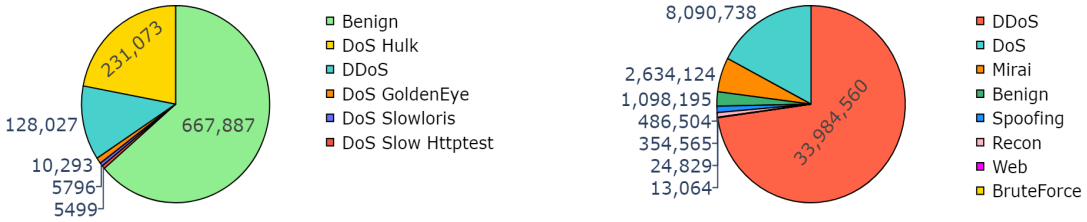
Experimental Design

7.1 Dataset Description

7.1.1 CICIDS2017 Dataset Details

CICIDS2017 benchmark dataset composed of real network attacks [233, 234] is employed for this research to train the IDS with typical attacks and benign observations. Fig. 7.1a depicts the proportion of each DoS/DDoS attack class and benign class in the dataset. The total percentages of DoS Hulk, DDoS, DoS GoldenEye, DoS Slowloris, and DoS SlowHttptest attack flows are 22.04%, 12.21%, 0.98%, 0.55% and 0.52% respectively. While the total percentage of benign flows in this dataset is 63.70%.

The dataset consists of 75 features. These features are generated by feeding the packet captures that resemble real-world data into the CICFlowmeter. A detailed explanation of all the features can be found in [8]. The IDS (Discriminator in our CVAE-AN model) is trained using CICIDS2017 data, with 75% randomly selected flows for training, and the remaining 25% randomly selected flows for testing against



(a) CICIDS2017 DDoS/DoS dataset details

(b) CICIoT2023 dataset details

Figure 7.1: The number of benign and attack observations for CICIDS2017 and CICIoT2023 datasets respectively.

Table 7.1: Best feature subset selected using HFSE Technique for CICIDS2017 dataset

S.No	Feature name	Chi2	RFE	LR	RFC	L-GBM	ETC	L-SVC	votes
1	Average packet size	✓	✓	✓	✓	✓	✓	✓	7
2	Idle Mean	✓	✓	✓	✓	✗	✓	✓	6
3	Packet Length Variance	✓	✓	✓	✓	✗	✓	✓	6
4	Max Packet Length	✓	✓	✓	✓	✗	✓	✓	6
5	Flow IAT Max	✓	✓	✓	✓	✓	✓	✗	6
6	Bwd Packet Length Std	✓	✓	✓	✓	✗	✓	✓	6
7	Bwd Packet Length Mean	✓	✓	✓	✓	✗	✓	✓	6
8	Avg Bwd Segment Size	✓	✓	✓	✓	✗	✓	✓	6
9	Init Win bytes forward	✓	✗	✗	✓	✓	✓	✓	5
10	Packet Length Std	✓	✓	✗	✓	✓	✓	✗	5
11	Packet Length Mean	✓	✓	✗	✓	✓	✓	✗	5
12	Init Win bytes backward	✓	✗	✓	✓	✓	✗	✓	5
13	Idle Max	✓	✓	✓	✗	✗	✓	✓	5
14	Fwd Packet Length Max	✗	✓	✓	✓	✓	✗	✓	5
15	Flow IAT Std	✓	✓	✓	✗	✓	✗	✓	5
16	Flow IAT Mean	✓	✓	✓	✗	✓	✗	✓	5
17	Bwd Packet Length Min	✓	✓	✓	✓	✗	✗	✓	5
18	Fwd IAT Total	✓	✗	✗	✗	✓	✓	✓	4
19	Bwd Packet Length Max	✓	✓	✗	✓	✗	✓	✗	4
20	Fwd Packet Length Mean	✗	✓	✓	✓	✓	✗	✗	4
21	Fwd IAT Std	✓	✗	✗	✓	✓	✓	✗	4
22	Fwd IAT Mean	✓	✗	✓	✗	✓	✗	✓	4
23	Fwd IAT Max	✓	✗	✗	✓	✓	✓	✗	4
24	Bwd Packets per second	✗	✓	✓	✗	✓	✗	✓	4

typical attacks as a conventional ML practice. Table 7.1 displays the feature subset chosen through the suggested HFSE technique for the CICIDS2017 dataset. The features are sorted according to their rank based on the number of votes by different feature selectors in the ensemble.

Table 7.2: Best feature subset selected using HFSE Technique for CICIoT2023 dataset

S.No	Feature name	Chi2	RFE	LR	RFC	L-GBM	ETC	L-SVC	votes
1	urg count	✓	✓	✓	✓	✓	✓	✓	7
2	rst count	✓	✓	✓	✓	✓	✓	✓	7
3	Header Length	✓	✓	✓	✓	✓	✓	✓	7
4	Magnitude	✓	✓	✓	✗	✓	✓	✓	6
5	Duration (TTL)	✓	✓	✗	✓	✓	✓	✓	6
6	Flow duration	✗	✓	✓	✓	✓	✗	✓	5
7	Variance	✓	✓	✗	✓	✗	✓	✓	5
8	Std	✓	✓	✓	✓	✗	✗	✓	5
9	Radius	✓	✓	✓	✗	✓	✗	✓	5
10	Max pkt length	✓	✓	✓	✗	✓	✗	✓	5
11	syn flag number	✓	✓	✓	✗	✗	✗	✓	4
12	rst flag number	✓	✓	✓	✗	✗	✗	✓	4
13	ack flag number	✓	✓	✗	✗	✗	✓	✓	4
14	ICMP	✓	✓	✓	✗	✗	✗	✓	4
15	syn count	✗	✓	✗	✗	✓	✗	✓	3
16	ack count	✗	✓	✓	✗	✗	✗	✓	3
17	Number	✗	✗	✗	✓	✗	✓	✓	3
18	IAT	✗	✗	✗	✓	✓	✓	✗	3
19	HTTPS	✓	✗	✗	✓	✗	✓	✗	3
20	fin flag number	✓	✓	✗	✗	✗	✗	✗	2

7.1.2 CICIoT2023 Dataset Details

To substantiate our findings, we also incorporate the most up-to-date attack dataset, namely, CICIoT2023 [9] for this research. This dataset consists of real-time data captured in an IoT environment with 105 devices. It comprises benign data and seven categories of attacks such as DDoS, DoS, Mirai, Spoofing, Reconnaissance, Web, and BruteForce. Fig. 7.1b shows the number of flows for different classes of attacks and benign data for the CICIoT2023 dataset. Since this dataset encompasses approximately 548GB of traffic information [9], we employ a smaller dataset (first 17 subsets of CICIoT2023) for training our model. The rest of the data subsets are used randomly for training the polymorphic attacker class-wise for each attack and evaluating the IDS against non-AI attacks. Table 7.2 displays the feature subset chosen through the suggested HFSE technique for the CICIoT2023 dataset.

The functional (red) and non-functional (green) feature subsets selected using SHAP for CICIDS2017 and CICIoT2023 datasets respectively are displayed in Fig. 7.2. The functional features are kept unmutated while synthesizing polymorphic

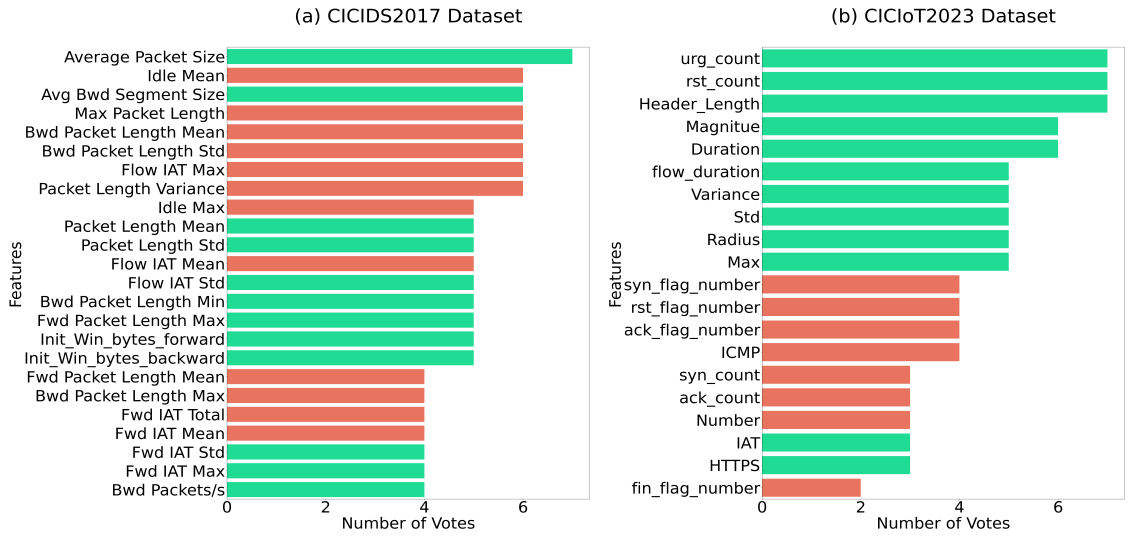


Figure 7.2: Feature subsets selected from CICIDS2017 and CICIoT2023 datasets by employing HFSE voting ensemble. Attack (functional) features identified using SHAP XAI are displayed in red color.

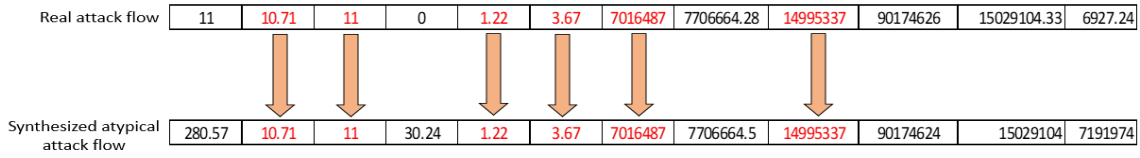


Figure 7.3: Preserving functional attack features when generating dynamically changing atypical/polymorphic attacks. Only the first 12 features are depicted here out of 24 selected features in total for CICIDS2017 data. Features depicted in red are functional attack features identified using SHAP. Notice that these features are kept constant when polymorphic attacks are generated to preserve the attack characteristics.

attacks to preserve the nature of the attack such that it resembles a real network attack. Fig. 7.3 depicts the process of preserving the functional attack characteristics while generating polymorphic attacks for the CICIDS2017 dataset.

Table 7.3: Polymorphic attacks (non-AI) for Slowloris DoS class. PA represents a polymorphic attack.

Polymorphic Attack	Duration	Number of Flows	p	s	ua
PA1 (non-AI)	3600 sec	10165	80	500	No
PA2 (non-AI)	10800 sec	33406	80	1000	No
PA3 (non-AI)	7200 sec	20536	80	5000	Yes

Table 7.4: Polymorphic attacks (non-AI) for Slow Httptest DoS class.

PA	Mode	Duration	# Flows	c	i	r	l	t	x	p	s	w	y	n	z	k
PA1 (non-AI)	H	2475 sec	8550	5000	25	500	10800	GET	24	10	-	-	-	-	-	-
PA2 (non-AI)	B	7200 sec	17185	5000	120	500	7200	POST	10	10	4096	-	-	-	-	-
PA3 (non-AI)	X	1800 sec	9205	10000	10	450	1800	GET	32	5	-	512	1024	5	32	3

7.2 Non-AI Polymorphic Attack Details

We employ several commonly available attack tools in a virtual network to generate non-AI polymorphic attacks. We alter these tool-based characteristics from their original settings to random values to create new feature profiles. For each attack class in the CICIDS2017 dataset, we synthesize three non-AI polymorphic attacks.

For launching the Slowloris DoS attack using Slowloris attack tool [249], we use target port number (p), number of sockets (s), and random user agents (ua) attributes to generate different feature profiles for each polymorphic attack. The details of polymorphic slowloris attacks with different feature profiles are given in Table 7.3.

To create different feature profiles for a Slow Httptest DoS attack, it is launched in 3 modes such as Slow Header (H), Slow Body (B), and Slow Read (X) using the commonly available DoS stress testing tool [232]. The attributes used for synthesizing different Slow Httptest DoS attacks are explained in detail in Table 5.2. Table 7.4 presents the details of polymorphic Slow Httptest DoS attack with different feature profiles.

Table 7.5: Polymorphic attacks (non-AI) for GoldenEye DoS class.

PA	Duration	# Flows	Useragents (u)	w	s	m	n	d
PA1 (non-AI)	600 sec	18996	randomly generated	10	500	GET	True	False
PA2 (non-AI)	600 sec	12084	randomly generated	50	1000	GET	True	False
PA3 (non-AI)	900 sec	13307	randomly generated	100	2000	GET	True	True

GoldenEye DoS attack is launched by utilizing the open source GoldenEye DoS testing tool [250]. The attributes for generating different feature profiles are user agents (u), number of concurrent workers (w), number of concurrent sockets (s), HTTP method ('get', 'post' or 'random') used (m), no SSL certificate verification (n), debug mode enabled (d). Table 7.5 provides the details of different feature profiles for polymorphic GoldenEye DoS attacks.

Hulk DoS attack is launched using the open source Hulk DoS tool [251]. The flooding attack is targeted at the web server in the virtual network to disrupt its normal functioning by prolonging its unavailability to legitimate users. The attack is launched for 3600 seconds with tool-based default attack parameters. During the attack, 5374897 requests are automatically sent from the attacker to the target server to overwhelm it. A substantial volume of packets is collected and data flows are isolated from it using the flow analyzer to synthesize three non-AI attack datasets.

To carry out non-AI synthesized DDoS attacks, we employ the open-source known as High Orbit Ion Cannon (HOIC) attack tool [252]. This tool is used to initiate numerous HTTP requests on the target web server within the virtual network, aiming to overwhelm the server and induce service disruption. The attack is launched for 3600 seconds with 20 threads and 5 simultaneous attacks. Three distinct non-AI attack datasets are extracted from the collected data packets, following a process similar to the extraction of Hulk DoS attacks.

7.3 Evaluation Metrics

The performance of IDS can be effectively measured if it can distinctly identify both attack and benign observations [45]. For example, when examining the results of an IDS if we focus on improving only attack detection performance and ignore benign detection, it could increase the number of false alarms for the NIDS. This flood of false triggers for a non-malicious activity within the network can deplete network resources such as memory, time, and processing power. As a result, genuine users experience delayed responses to real network incidents.

Given below are the standard and applied evaluation metrics used for the analysis of the IDS. These metrics reflect the performance of the IDS against both attack and benign traffic.

- **True Positive Rate (TPR)**: It is the rate of correctly identified attack observations with respect to all the attack observations in the data. It is also commonly known as recall or sensitivity.

$$TPR = \frac{TP}{(TP + FN)} \tag{7.1}$$

- **True Negative Rate (TNR)**: It is the rate of correctly identified benign observations with respect to all the benign observations in the data. It is also commonly known as specificity or selectivity.

$$TNR = \frac{TN}{(TN + FP)} \tag{7.2}$$

- **Balanced Accuracy (BA)**: This metric is especially beneficial when the classes in the input data are imbalanced. BA takes into account both attack

and benign samples identified. It is measured by the mean of TPR and TNR for a binary IDS.

$$BA = \frac{(TPR + TNR)}{2} \tag{7.3}$$

- **IDS Proficiency (pID)**: This metric computes the efficiency of an IDS for the detection of both attack and benign observations. pID considers both TPR and TNR values for an IDS and the range for this metric is from -1 to +1. Here, -1 signifies that the IDS cannot identify any attack or benign samples and vice versa for +1.

$$pID = TPR + TNR - 1 \tag{7.4}$$

- **Overall Error Rate (OER)**: This measures the total erroneous benign or attack classifications done by the IDS model.

$$OER = 1 - BA \tag{7.5}$$

To assess the success of the attacker in eluding IDS, the following metric is introduced:

- **Evasion Success Rate (ESR)**: This measures the rate of increase of undetected atypical attack observations by the IDS. It is also known as the miss rate.

$$ESR = \frac{FN}{(FN + TP)} = 1 - TPR \tag{7.6}$$

Here, TP, TN, FP, and FN represent True Positives, True Negatives, False Positives, and False Negatives respectively.

Chapter 8

Performance Evaluation

In this chapter, we explain and analyze the results obtained from various experiments against typical and dynamically changing atypical/polymorphic attacks using our technique.

8.1 Model Structure

For this research, we adopt the Keras library and Tensorflow platform to build the CVAE-AN model. The attack generator is a CVAE model consisting of encoder and decoder networks. The stacked discriminator network with independent logical supervised and unsupervised models having shared weights is based on the design in [29]. The supervised discriminator acts as an IDS classifier with a softmax activation. The unsupervised discriminator distinguishes real input data from synthesized attack data with a binary activation.

The summary of configuration parameters of CVAE-AN for the CICIDS2017 dataset is obtained from [3] through multiple training iterations and refinement. Table 8.1 summarizes the hyperparameter optimization and best configuration pa-

Table 8.1: Hyperparameter Optimization and best configuration parameters for CVAE-AN with CICIDS2017 dataset.

Hyperparameter	**Search Space**	**Best value**	**Results on TA**	**Results on Adv Shttp PA1**
Number of neurons in input layer of the encoder	[20, 24, 76]	24		
Number of layers in the encoder, decoder, and discriminator	[2-4]	2		
Number of neurons in encoder's hidden layers	[76-76-76, 128-128-128, 128-128]	128-128		
Number of neurons in latent layer	[30, 5]	5		
Number of neurons in decoder's hidden layers	[76-76-76, 128-128-128, 128-128]	128-128	BA= 93.04% pID = +0.86 OER= 6.96% ESR= 1.58%	BA= 43.49% pID = -0.13 OER=56.51% **ESR= 99.96%**
Number of neurons in output layer of the decoder	[20, 24, 76]	24		
Number of neurons in Discriminator's hidden layers	[128-128-128, 128-128]	128-128		
Activation function in Discriminator's hidden layers	LeakyRelu (alpha=0.2)	LeakyRelu (alpha=0.2)		
Dropout	0.4	0.4		
Optimizer	Adam (beta1=0.5)	Adam (beta1=0.5)		
Learning rate	[0.0001-0.0004]	0.0002		

rameters employed for the CVAE-AN model with the CICIDS2017 dataset. Table 8.2 provides the specifics of hyperparameter optimization and identification of best IDS configuration values for the CICIoT2023 dataset. The initial goal is to have polymorphic/atypical attacks to evade detection by the IDS. Therefore, we choose the set of hyperparameters that lead to the highest Evasion Success Rate (ESR) for a polymorphic/atypical attack. Afterward, we employ these selected hyperparameters to build our IDS for subsequent training against other polymorphic/ atypical attack classes.

8.2 Result Analysis

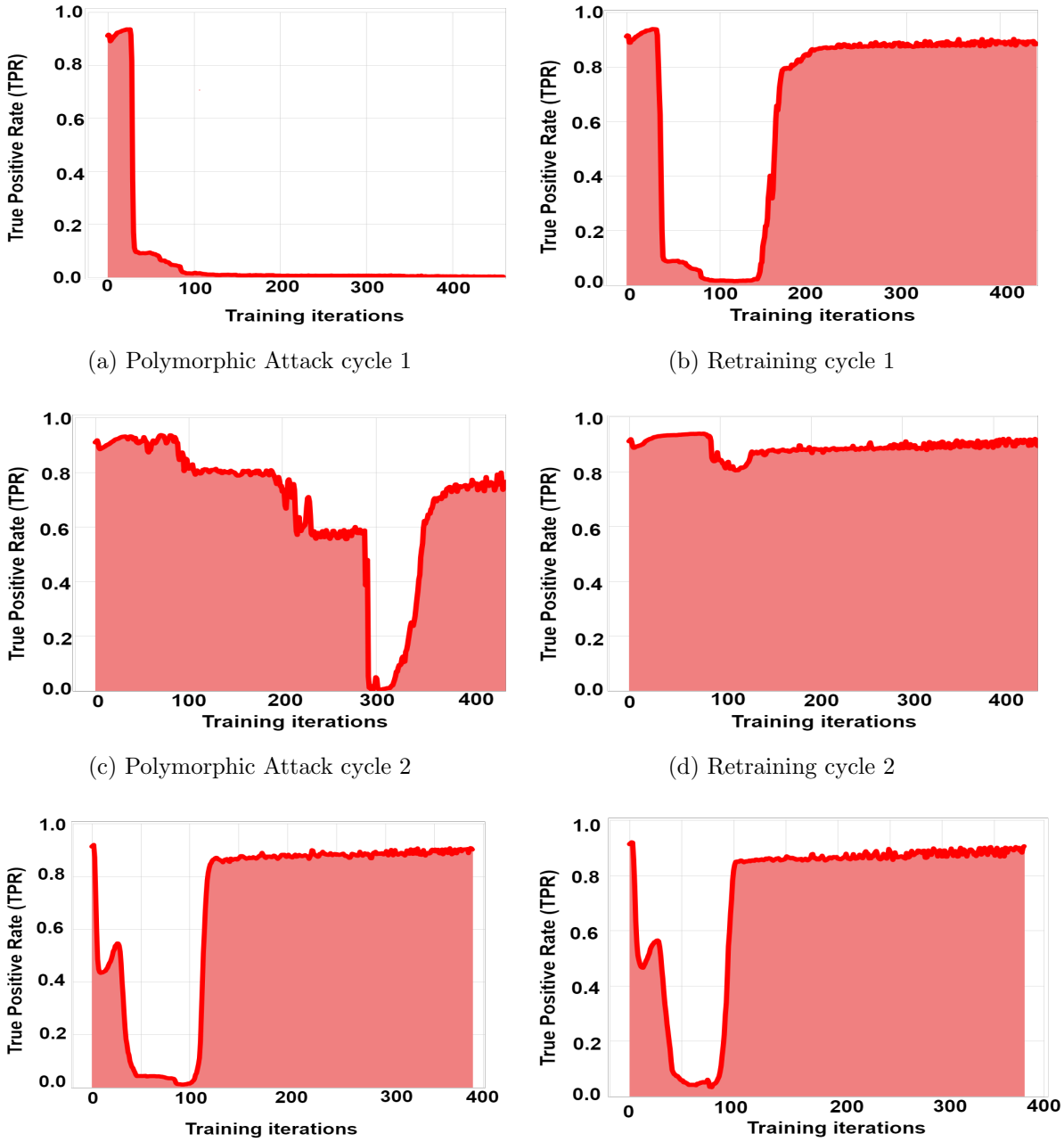
Multiple performance metrics are used to evaluate the results of the IDS on typical attacks and dynamically changing atypical/polymorphic attacks. We represent at-

Table 8.2: Hyperparameter Optimization and best configuration parameters for CVAE-AN with CICIoT2023 dataset.

Hyperparameter	**Search Space**	**Best value**	**Results on TA**	**Results on Adv Recon PA1**
Number of neurons in input layer	20	20		
Number of layers in the encoder, decoder, and discriminator	[2-4]	3		
Number of neurons in encoder's hidden layers	[1024-512-256-128, 512-256-128, 256-128-128, 128-128-64, 128-128]	128-128-64		
Number of neurons in latent layer	10	10		
Number of neurons in decoder's hidden layers	[128-256-512-1024, 128-256-512, 128-128-256, 64-128-128, 128-128]	64-128-128	BA= 98.87% pID = +0.98 OER= 1.13% ESR= 0.02%	BA= 57.31% pID = +0.15 OER= 42.69% **ESR= 85.27%**
Number of neurons in output layer	20	20		
Number of neurons in Discriminator's hidden layers	[1024-512-256-128, 512-256-128, 256-128-128, 128-128-64, 128-128]	128-128-64		
Activation function in Discriminator's hidden layers	LeakyRelu (alpha=0.2)	LeakyRelu (alpha=0.2)		
Dropout	0.4	0.4		
Optimizer	Adam (beta1=0.5)	Adam (beta1=0.5)		
Learning rate	[0.0001-0.0007, 0.001]	0.0005		
Batch size	[32, 64]	32		
Number of epochs	500	500		

tacks in the hold-out CICIDS2017 and CICIoT2023 data as typical attacks because these attacks are known to the IDS. Our CVAE attacker can only synthesize polymorphic attacks for a single class at a time. We assume that the polymorphic attacker modifies the feature profile to launch atypical attacks each time the IDS can successfully identify the attack. The polymorphic attack cycles are followed by incremental adversarial training of the IDS to improve its performance.

Fig. 8.1 depicts the changes in TPR of our IDS when a Slow Httptest DoS attack profile is mutated by the polymorphic CVAE attacker for each polymorphic attack cycle. From Fig. 8.1a, during the first polymorphic attack cycle, the IDS can't identify any attacks, therefore the TPR is reduced to almost zero. An improvement in the TPR results is depicted after the first adversarial retraining phase in Fig.



(a) Polymorphic Attack cycle 1

(b) Retraining cycle 1

(c) Polymorphic Attack cycle 2

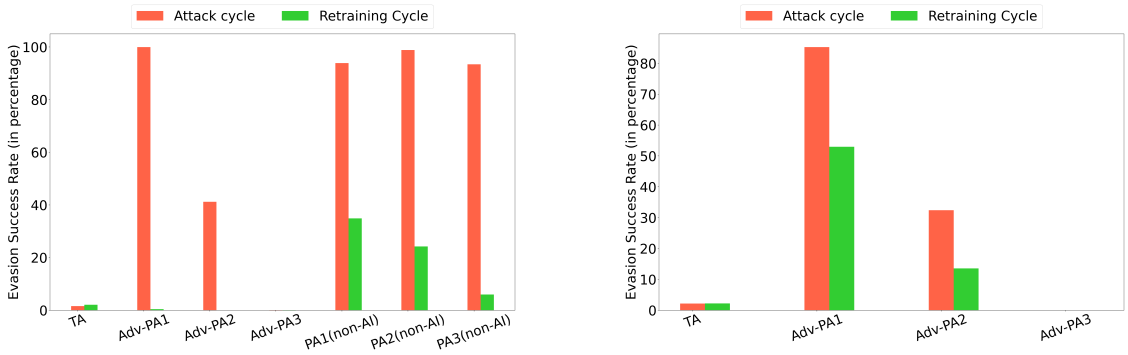
(d) Retraining cycle 2

(e) Polymorphic Attack cycle 3

(f) Retraining cycle 3

Figure 8.1: Effect on TPR of the IDS when a Slow Httptest DoS attack profile (from CICIDS2017 dataset) is mutated by the polymorphic attack generator.

8.1b. During the second polymorphic attack cycle, the polymorphic attack generator changes the feature profile again intending to evade detection by the IDS. As seen



(a) CICIDS2017 dataset (Slow Httptest DoS attack) (b) CICIoT2023 dataset (Recon attack)

Figure 8.2: Comparative analysis of Evasion Success Rate for Slow Httptest DoS attack from CICIDS2017 dataset and Recon attack from CICIoT2023 dataset. Here TA represents typical attacks, Adv-PA symbolizes adversarial polymorphic attacks and PA (non-AI) symbolizes non-AI polymorphic attacks.

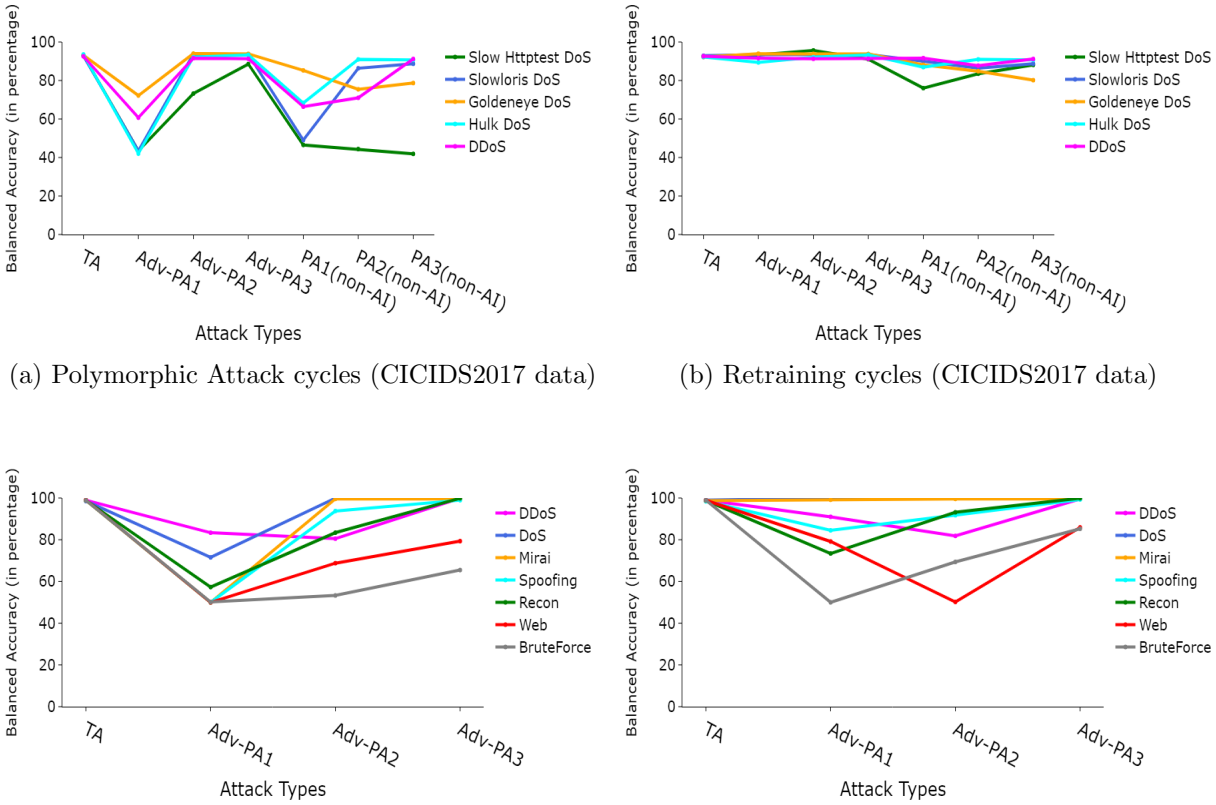
in Fig. 8.1c, the TPR results are reduced again to approximately 75% which means that the IDS is misclassifying 25% polymorphic attacks. To enhance the overall effectiveness of the IDS, we again perform adversarial retraining. The polymorphic attacker keeps on launching atypical attacks on the victim IDS until it exhausts all of its features or until it is capable of recognizing all the new attacks. In this case, as seen in Fig. 8.1e, initially during the training process, the TPR for IDS is reduced, but it gradually learns the pattern of polymorphic attacks after multiple retraining sessions and thus can identify the attack with approximately 90% TPR both during the attack phase as well as retraining phase.

To evaluate the success of the polymorphic attacker, we employ the Evasion Success Rate (ESR) metric. Fig. 8.2a and Fig. 8.2b compare ESRs for Slow Httptest DoS attack and Recon attack from CICIDS2017 and CICIoT2023 datasets respectively. As shown in the figure, the ESR metric value for typical attacks is very low as compared to other polymorphic attacks. Each polymorphic attack cycle represents a successful evasion of IDS by the polymorphic attacker. After incremental retraining,

the ESR reduces to very low values in most of the cases indicating that the IDS can identify the attack. Although for non-AI synthesized polymorphic attacks, the ESR is higher (even after adversarial training), it gradually reduces to a lower value after the third retraining cycle. We randomly selected one attack from each dataset for this analysis. Similar results are obtained using other attack classes and as such have not been discussed here.

We employ the balanced accuracy metric to offer a comprehensive assessment of the IDS model's effectiveness in detecting both attack and benign observations. This metric considers both true positive rate (TPR) and true negative rate (TNR), which is particularly crucial when dealing with an imbalanced dataset with minority attack samples. We believe that this metric provides a fairer representation of the performance of our IDS model across different classes in the input data. Fig. 8.3 displays the balanced accuracy values obtained by the CVAE-AN IDS model for different classes of polymorphic attack scenarios and subsequent retraining cycles. Each point on the plot corresponds to the balanced accuracy value of our IDS for each attack cycle in the polymorphic attack chain.

As shown in Fig. 8.3a, the balanced accuracy for typical attacks is high for all the attack classes. But during the first polymorphic attack (PA1) cycle against the IDS, the balanced accuracy is reduced to 40% for the Slow Httptest DoS attack in the CICIDS2017 dataset. This is because the TPR or the number of correctly identified attacks by the IDS is reduced during the polymorphic attack cycle. Similar performance can be seen in Fig. 8.3c for the CICIoT2023 dataset. However, the IDS shows improvement in its results after the first incremental adversarial training cycle as shown in Fig. 8.3b and Fig. 8.3d. While the polymorphic attacker keeps on launching atypical attacks to evade detection in the subsequent attack cycles. The IDS continuously keeps on updating itself through incremental training cycles until



(a) Polymorphic Attack cycles (CICIDS2017 data)

(b) Retraining cycles (CICIDS2017 data)

(c) Polymorphic Attack cycles (CICIoT2023 data)

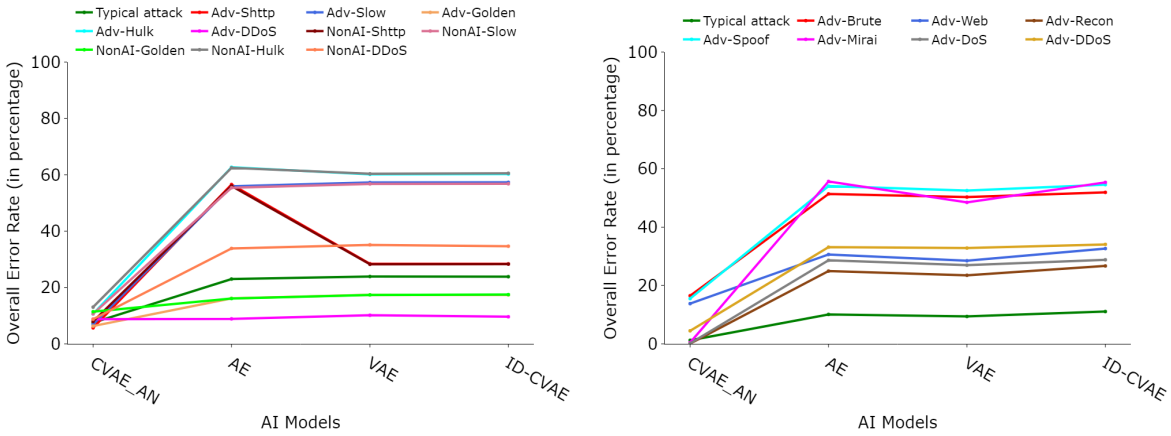
(d) Retraining cycles (CICIoT2023 data)

Figure 8.3: Comparison of Balanced Accuracy for the CVAE-AN IDS (polymorphic attack and retraining cycles). Here TA symbolizes typical attacks from the set-aside CICIDS2017 and CICIoT2023 data, Adv-PA symbolizes Adversarial polymorphic attacks and PA (non-AI) symbolizes non-AI polymorphic attacks.

the polymorphic attacker runs out of features and can no longer evade detection. Here, we employ one attack class for generating polymorphic attacks, followed by another attack class, and so on until all the attacks can be identified by the IDS.

8.3 Comparison with State-of-the-Art Techniques

To emphasize the effectiveness of our CVAE-AN IDS, its performance is compared with other anomaly detection techniques such as Autoencoder (AE) [5], Variational



(a) CICIDS2017 data (b) CICIoT2023 data

Figure 8.4: Analysis of Overall Error Rate (OER) for polymorphic attack-1 (PA1) across multiple attack classes within the CICIDS2017 and CICIoT2023 datasets.

Autoencoder (VAE) [5], and Conditional Variational Autoencoder IDS (ID-CVAE) [253]. In this research context, the OER metric provides the total error in misclassification for both attack and benign instances. By using this metric, our goal is to assess the overall performance of our IDS in identifying both attack and benign observations, considering the misclassification made by the system. We aim to demonstrate that our IDS outperforms other state-of-the-art anomaly detection techniques by making fewer misclassifications.

In Fig. 8.4, a comparison of the Overall Error Rate (OER) in identifying polymorphic attacks is presented, showcasing the performance of CVAE-AN IDS against various state-of-the-art anomaly detection models. The results presented are specific to one polymorphic attack, and it is important to note that similar performances are observed across other cycles of polymorphic attacks. The noteworthy aspect is the consistent performance of the proposed IDS across multiple cycles of polymorphic attacks, demonstrating stable outcomes. Significantly lower OER values achieved by the CVAE-AN IDS in comparison to other anomaly detection models indicate its

Table 8.3: Comparing the performance of the CVAE-AN IDS with current network anomaly detection techniques for the CICIDS2017 dataset. Here Adv-PA symbolizes Adversarial polymorphic attacks, PA (non-AI) symbolizes non-AI polymorphic attacks, pID symbolizes IDS Proficiency, and ESR symbolizes Evasion Success Rate.

Attack Class	**Attack Type**	**CVAE-AN**		**AE [5]**		**VAE [5]**		**ID-CVAE [253]**	
		pID	ESR	pID	ESR	pID	ESR	pID	ESR
CICIDS2017 test	TA	**+0.84**	**1.43%**	+0.54	33.89%	+0.52	32.93%	+0.52	32.92%
Slowloris DoS	Adv-PA1	**+0.86**	**0.35%**	-0.12	99.47%	-0.15	99.38%	-0.15	99.37%
	Adv-PA2	**+0.90**	**0.35%**	-0.08	99.48%	-0.10	99.37%	-0.10	99.37%
	Adv-PA3	**+0.76**	**0.35%**	-0.29	99.48%	-0.28	99.38%	-0.29	99.37%
	PA1 (non-AI)	**+0.79**	**7.87%**	-0.11	98.46%	-0.14	98.33%	-0.14	98.33%
	PA2 (non-AI)	**+0.70**	**7.51%**	-0.39	98.62%	-0.37	98.16%	-0.38	98.13%
	PA3 (non-AI)	**+0.75**	**5.28%**	-0.40	99.53%	-0.39	99.43%	-0.40	99.42%
Slow Httptest DoS	Adv-PA1	**+0.89**	**0.37%**	-0.13	99.77%	+0.43	40.71%	+0.43	40.56%
	Adv-PA2	**+0.84**	**0.02%**	+0.67	5.16%	+0.34	39.48%	+0.33	39.45%
	Adv-PA3	**+0.85**	**0.08%**	-0.24	99.80%	-0.24	99.78%	-0.24	99.77%
	PA1 (non-AI)	**+0.84**	**4.94%**	-0.12	98.71%	+0.44	40.35%	+0.43	40.39%
	PA2 (non-AI)	**+0.67**	**17.48%**	+0.33	39.37%	+0.34	39.30%	+0.34	39.30%
	PA3 (non-AI)	**+0.79**	**5.99%**	-0.24	99.34%	-0.24	99.45%	-0.24	99.42%
GoldenEye DoS	Adv-PA1	**+0.87**	**0.37%**	+0.68	21.73%	+0.65	21.76%	+0.65	21.63%
	Adv-PA2	**+0.82**	**0.34%**	+0.47	22.15%	+0.48	21.95%	+0.48	21.82%
	Adv-PA3	**+0.85**	**0.25%**	+0.33	22.08%	+0.35	21.88%	+0.35	21.74%
	PA1 (non-AI)	**+0.77**	**10.58%**	+0.68	21.62%	+0.65	21.77%	+0.65	21.83%
	PA2 (non-AI)	**+0.70**	**17.64%**	+0.67	19.73%	+0.66	19.02%	+0.66	19.01%
	PA3 (non-AI)	**+0.60**	20.70%	+0.44	21.06%	+0.49	**17.94%**	+0.48	**17.94%**
Hulk DoS	Adv-PA1	**+0.79**	**0.09%**	-0.25	95.60%	-0.20	91.10%	-0.21	90.76%
	Adv-PA2	**+0.84**	**0.10%**	-0.11	95.92%	-0.10	91.52%	-0.10	91.15%
	Adv-PA3	**+0.84**	**0.10%**	-0.09	95.92%	-0.11	91.55%	-0.11	91.12%
	PA1 (non-AI)	**+0.74**	**4.92%**	-0.25	95.17%	-0.21	91.44%	-0.21	91.28%
	PA2 (non-AI)	**+0.82**	**1.69%**	-0.07	92.29%	-0.09	90.10%	-0.09	90.10%
	PA3 (non-AI)	**+0.81**	**2.97%**	-0.02	88.49%	-0.03	84.11%	-0.04	84.02%
DDoS	Adv-PA1	**+0.83**	0.01%	+0.82	**0.00%**	+0.80	**0.00%**	+0.81	**0.00%**
	Adv-PA2	**+0.75**	0.01%	+0.60	**0.00%**	+0.59	**0.00%**	+0.61	**0.00%**
	Adv-PA3	+0.82	0.01%	**+0.84**	**0.00%**	+0.81	**0.00%**	+0.83	**0.00%**
	PA1 (non-AI)	**+0.83**	**0.01%**	+0.32	50.00%	+0.30	50.00%	+0.31	50.00%
	PA2 (non-AI)	**+0.75**	**0.04%**	+0.10	50.02%	+0.09	50.00%	+0.11	50.00%
	PA3 (non-AI)	**+0.82**	**0.06%**	+0.34	50.00%	+0.31	50.00%	+0.33	50.00%

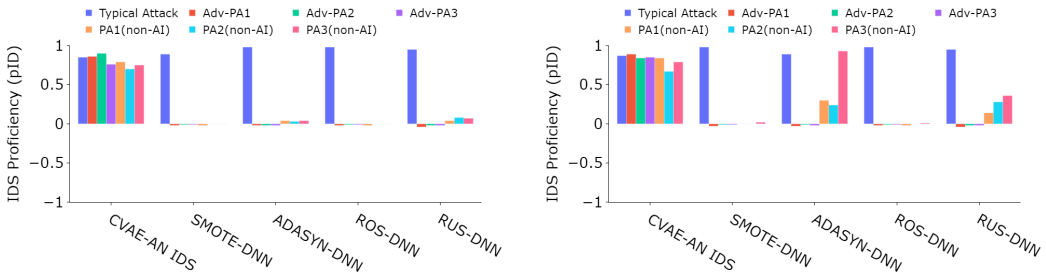
remarkable performance in minimizing misclassifications. This effectiveness extends to detecting both atypical/polymorphic attacks and benign samples, distinguishing it from other anomaly detection models.

Table 8.3 and Table 8.4 provide a comparison of our network intrusion detection technique against state-of-the-art network anomaly detection techniques using CICIDS2017 and CICIoT2023 datasets respectively. We aim to demonstrate the robustness of our model in detecting dynamically evolving atypical/polymorphic attacks, as compared to the current anomaly detection methods used for recognizing

Table 8.4: Comparing the performance of the CVAE-AN IDS with current network anomaly detection techniques for the CICIoT2023 dataset. Here Adv-PA symbolizes Adversarial polymorphic attacks, pID symbolizes IDS Proficiency, and ESR symbolizes Evasion Success Rate.

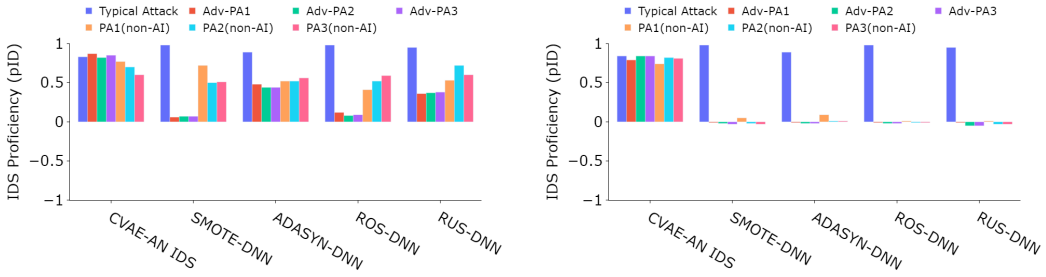
Attack Class	**Attack Type**	**CVAE-AN**		**AE [5]**		**VAE [5]**		**ID-CVAE [253]**	
		pID	ESR	pID	ESR	pID	ESR	pID	ESR
CICIoT2023 test	TA	**+0.96**	**1.85%**	+0.80	9.21%	+0.81	6.32%	+0.78	9.91%
Brute Force	Adv-PA1	**+0.67**	**31.29%**	-0.03	91.21%	-0.01	89.24%	-0.04	91.02%
	Adv-PA2	**+0.98**	**0.00%**	+0.30	59.13%	+0.35	52.94%	+0.28	61.83%
	Adv-PA3	**+0.71**	**27.34%**	-0.02	91.73%	+0.02	86.24%	-0.04	91.28%
Web	Adv-PA1	**+0.72**	**27.32%**	+0.39	49.80%	+0.43	43.87%	+0.35	52.89%
	Adv-PA2	**+0.73**	**26.54%**	+0.37	51.76%	+0.42	46.16%	+0.33	54.47%
	Adv-PA3	**+0.72**	**28.02%**	+0.38	51.84%	+0.40	46.31%	+0.32	55.35%
Reconnaissance	Adv-PA1	**+0.95**	**4.90%**	+0.50	39.01%	+0.53	34.26%	+0.47	41.12%
	Adv-PA2	**+0.95**	**4.91%**	+0.23	65.76%	+0.24	63.63%	+0.23	65.52%
	Adv-PA3	**+1.00**	**0.00%**	+0.24	65.18%	+0.26	61.70%	+0.23	65.05%
Spoofing	Adv-PA1	**+0.69**	**30.79%**	-0.08	97.22%	-0.05	92.65%	-0.09	96.76%
	Adv-PA2	**+0.83**	**16.20%**	-0.08	97.32%	-0.05	92.54%	-0.09	96.98%
	Adv-PA3	**+0.99**	**1.20%**	+0.23	66.21%	+0.29	58.95%	+0.20	68.62%
Mirai	Adv-PA1	**+0.99**	**0.75%**	-0.11	99.89%	+0.03	84.85%	-0.11	98.56%
	Adv-PA2	**+1.00**	**0.28%**	-0.10	98.76%	+0.68	19.15%	+0.39	49.16%
	Adv-PA3	**+0.99**	**0.85%**	-0.11	99.92%	-0.06	93.99%	-0.13	99.77%
DoS	Adv-PA1	**+1.00**	**0.01%**	+0.43	45.88%	+0.46	41.14%	+0.42	45.53%
	Adv-PA2	**+1.00**	**0.00%**	+0.42	47.12%	+0.46	42.21%	+0.41	46.74%
	Adv-PA3	**+1.00**	**0.00%**	+0.43	46.76%	+0.46	42.08%	+0.42	46.16%
DDoS	Adv-PA1	**+0.91**	**8.77%**	+0.34	55.09%	+0.34	52.19%	+0.32	55.48%
	Adv-PA2	**+0.99**	**0.96%**	+0.34	55.06%	+0.35	52.87%	+0.32	55.95%
	Adv-PA3	**+0.99**	**0.65%**	+0.30	58.52%	+0.27	59.45%	+0.26	61.15%

previously unseen attacks. Our IDS is trained for three incremental learning iterations against three adversarial polymorphic attacks and three non-AI polymorphic attacks for each attack class. The results specify that the proposed IDS can identify both typical/polymorphic attacks and benign observations which can be determined by observing the pID metric values when comparing other anomaly detection techniques. While lower pID values for AE, VAE, and ID-CVAE indicate their performances degrade when exposed to dynamically changing atypical attacks/polymorphic attacks. Higher ESR values suggest that the polymorphic attacker is successful in eluding detection by the IDS.



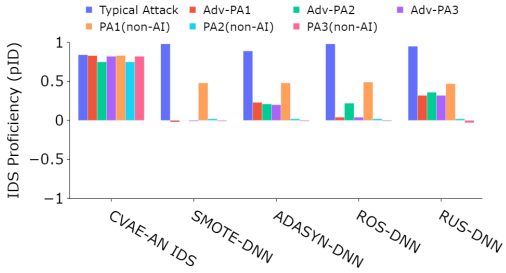
(a) Slowloris DoS poly attacks

(b) Slow Httptest DoS poly attacks



(c) GoldenEye DoS poly attacks

(d) Hulk DoS poly attacks

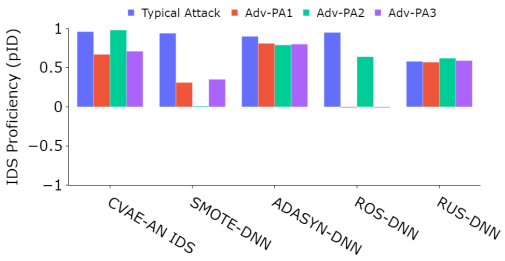


(e) DDoS poly attacks

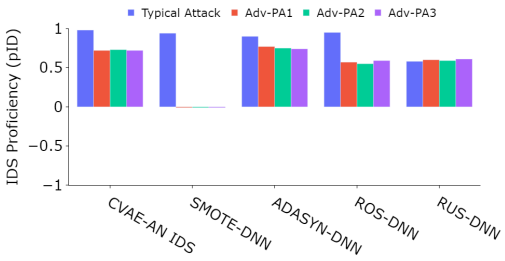
Figure 8.5: Comparison of proficiency for IDS trained using our incremental learning technique with a DNN trained using alternative class balancing methods on the CICIDS2017 dataset. Here Adv-PA symbolizes Adversarial polymorphic attacks, and PA (non-AI) symbolizes non-AI polymorphic attacks.

8.4 Comparing Class Balancing Techniques

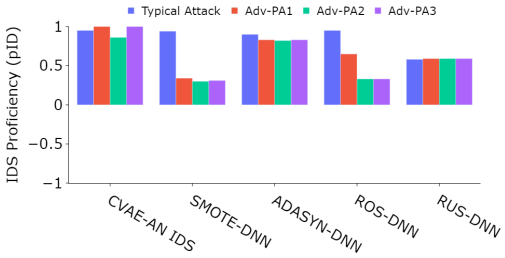
We employ the IDS proficiency (pID) metric to compute the efficiency of our IDS in detecting both attack and benign observations, particularly in the context of dealing with imbalanced datasets. Fig. 8.5 and Fig. 8.6 present a comparative analysis of the pID of CVAE-AN IDS trained using our incremental learning technique in contrast



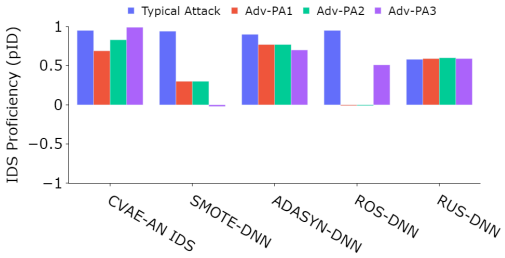
(a) Brute Force poly attacks



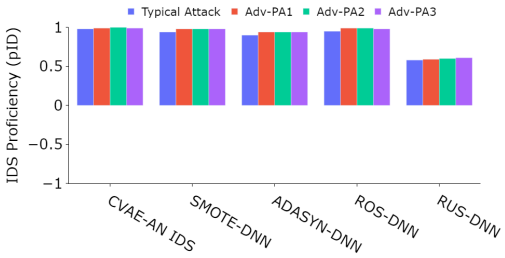
(b) Web poly attacks



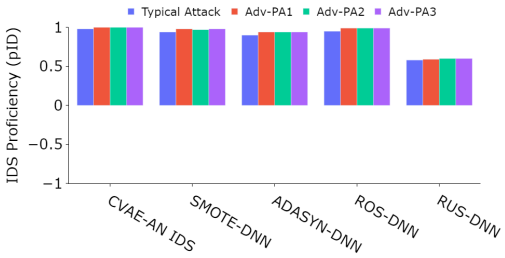
(c) Recon poly attacks



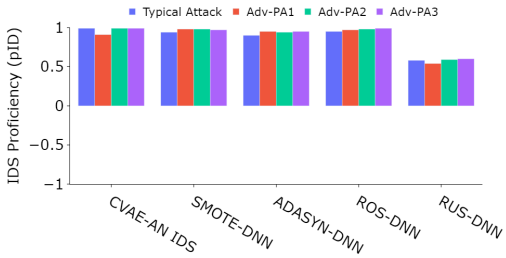
(d) Spoof poly attacks



(e) Mirai poly attacks



(f) DoS poly attacks



(g) DDoS poly attacks

Figure 8.6: Comparison of proficiency for IDS trained using our incremental learning technique with a DNN trained using alternative class balancing methods on the CICIoT2023 dataset. Here Adv-PA symbolizes Adversarial polymorphic attacks.

to a DNN trained on data which is resampled using several class balancing methods. The evaluation is conducted on datasets CICIDS2017 and CICIoT2023 respectively. The evaluation of performance considers various classes and types of dynamically changing atypical attacks/polymorphic attacks. From the Fig. 8.5 and Fig. 8.6, both the CVAE-AN IDS and DNN demonstrate the ability to identify typical attacks effectively, as indicated by higher pID values. However, when exposed to polymorphic attacks, the CVAE-AN IDS outperforms a standard DNN model significantly for most of the attacks in the CICIDS2017 and CICIoT2023 datasets. This highlights the resilience of our CVAE-AN IDS in the face of dynamically changing atypical attacks/ polymorphic attacks, illustrating its superiority in handling such challenging scenarios.

8.5 Polymorphic Attack Quality Analysis

In this section, we provide an analysis of the quality of adversarial atypical/polymorphic attacks generated by our system. To investigate atypical/polymorphic adversarial attack realism, we employ several techniques such as syntactic validation, hypothesis testing, statistical distance based analysis, and correlation analysis. These methods facilitate a comparison between the distributions of real and synthesized atypical/polymorphic attacks. Fig. 8.7 compares a few selected features for an original Slowloris DoS attack and a CVAE-AN synthesized adversarial attack for the same attack class. There is a lack of metrics that can measure this similarity of synthesized attacks with real attacks for network intrusion detection applications. Our aim through this research is to fill that gap to improve the detection of synthesized adversarial atypical/polymorphic attacks.

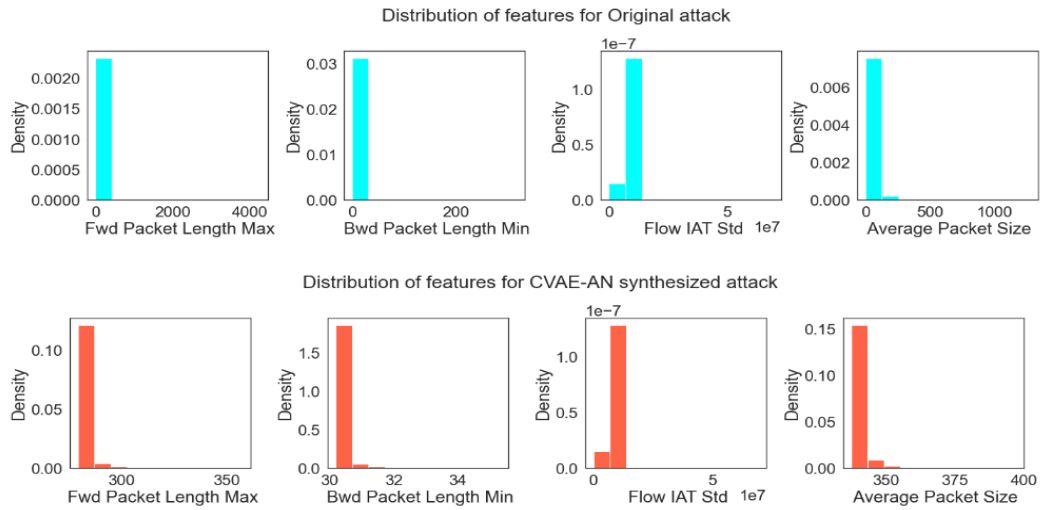


Figure 8.7: Comparison between the distributions of an original Slowloris DoS attack and a CVAE-AN synthesized adversarial polymorphic attack for the CICIDS2017 dataset.

8.5.1 Hypothesis testing

We employ Python's *scipy.stats* library and *ks_2samp* function to measure the KS statistic and p-value for comparing two data distributions. Since the KS test cannot be applied to test all the features in both datasets simultaneously, we compare each feature separately for the real attack and adversarial attack datasets. We then report the average KS-test statistic and average p-value over the entire dataset. Our experiments focus on adversarial polymorphic attacks (Adv-PA1) generated by our model CVAE-AN [3]. Tables 8.5 and 8.6 provide an analysis of the quality of polymorphic adversarial network attacks using multiple metrics on the CICIDS2017 and CICIoT2023 datasets, respectively. For the KS-test, we observe that for all the adversarial polymorphic attacks the average p-values are higher than the threshold value of 0.05, indicating that the adversarial attack data follows the same continuous distribution as real attack data.

Table 8.5: Analysis of the quality of polymorphic adversarial network attacks using multiple statistical techniques on the CICIDS2017 dataset. The table shows the average values for all the metrics.

Polymorphic attack class	KS Statistic	p-value	Hellinger Distance	KLD	JSD
Slowloris DoS Adv-PA1	0.52	0.42	0.24	0.79	0.13
Slowloris DoS Adv-PA2	0.52	0.42	0.24	0.79	0.13
Slowloris DoS Adv-PA3	0.52	0.42	0.24	0.79	0.13
Slow Httptest DoS Adv-PA1	0.41	0.50	0.26	0.73	0.13
Slow Httptest DoS Adv-PA2	0.44	0.50	0.29	1.07	0.16
Slow Httptest DoS Adv-PA3	0.45	0.46	0.34	1.23	0.22
GoldenEye DoS Adv-PA1	0.33	0.46	0.26	0.85	0.13
GoldenEye DoS Adv-PA2	0.34	0.46	0.26	0.85	0.13
GoldenEye DoS Adv-PA3	0.33	0.46	0.26	0.85	0.13
Hulk DoS Adv-PA1	0.42	0.42	0.25	0.56	0.13
Hulk DoS Adv-PA2	0.42	0.42	0.25	0.56	0.13
Hulk DoS Adv-PA3	0.42	0.42	0.25	0.56	0.13
DDoS Adv-PA1	0.39	0.44	0.24	0.73	0.11
DDoS Adv-PA2	0.39	0.44	0.24	0.73	0.11
DDoS Adv-PA3	0.39	0.44	0.24	0.72	0.11

Table 8.6: Analysis of the quality of polymorphic adversarial network attacks using multiple statistical techniques on the CICIoT2023 dataset. The table shows the average values for all the metrics.

Polymorphic attack class	KS Statistic	p-value	Hellinger Distance	KLD	JSD
Brute Force Adv-PA1	0.44	0.38	0.25	0.50	0.11
Brute Force Adv-PA2	0.42	0.37	0.26	0.52	0.11
Brute Force Adv-PA3	0.44	0.39	0.25	0.50	0.11
Web Adv-PA1	0.44	0.35	0.26	0.49	0.11
Web Adv-PA2	0.43	0.35	0.26	0.48	0.11
Web Adv-PA3	0.43	0.35	0.26	0.49	0.11
Reconnaissance Adv-PA1	0.39	0.35	0.29	0.61	0.13
Reconnaissance Adv-PA2	0.46	0.35	0.29	0.62	0.14
Reconnaissance Adv-PA3	0.47	0.35	0.29	0.62	0.13
Spoof Adv-PA1	0.40	0.35	0.26	0.48	0.11
Spoof Adv-PA2	0.38	0.35	0.26	0.49	0.11
Spoof Adv-PA3	0.40	0.35	0.26	0.49	0.11
Mirai Adv-PA1	0.59	0.38	0.33	1.55	0.20
Mirai Adv-PA2	0.56	0.38	0.33	1.44	0.20
Mirai Adv-PA3	0.59	0.36	0.32	1.44	0.19
DoS Adv-PA1	0.50	0.35	0.23	0.60	0.10
DoS Adv-PA2	0.34	0.25	0.23	0.58	0.10
DoS Adv-PA3	0.50	0.35	0.23	0.57	0.10
DDoS Adv-PA1	0.48	0.41	0.24	0.67	0.11
DDoS Adv-PA2	0.41	0.41	0.25	0.84	0.12
DDoS Adv-PA3	0.54	0.35	0.22	0.61	0.10

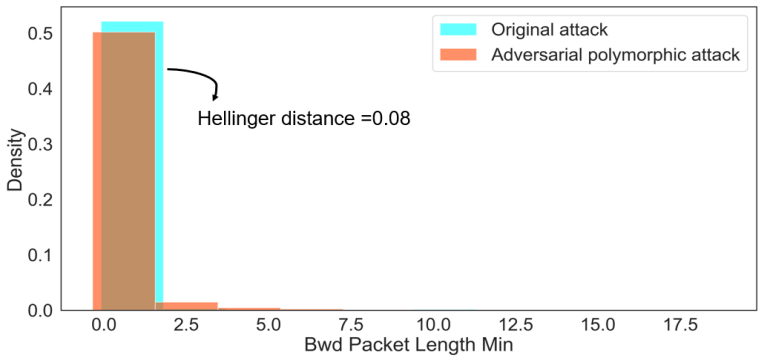


Figure 8.8: Comparison between the distributions of a real Slowloris DoS attack and a CVAE-AN synthesized adversarial polymorphic attack for the feature '*Bwd Packet Length Min*' based on Hellinger distance.

8.5.2 Statistical distance based analysis

Hellinger Distance

We apply a statistical distance based metric, Hellinger distance to analyze the similarity between the probability distribution of real attack data and adversarially synthesized atypical/polymorphic attacks. Similar to the KS-test analysis, we compare the distributions of individual feature values of the real attack data and adversarial atypical/polymorphic data separately to find the distance between them and then calculate the overall average distance to determine their similarity. For instance in Fig. 8.8, we compare the distribution of the feature '*Bwd Packet Length Min*' for a real Slowloris DoS attack from the CICIDS2017 dataset and the corresponding adversarial polymorphic attack synthesized using CVAE-AN. The calculated value for Hellinger distance for this input feature is 0.08 which is closer to 0 indicating that both these distributions have close resemblance to each other. This process is repeated for other attack features and the total average distance is computed.

From Tables 8.5 and 8.6, we observe that for most of the adversarial polymorphic attacks, the Hellinger distance value is closer to 0 indicating their close resemblance

to real network attacks.

Kullback-Leibler Divergence (KLD)

We apply KLD to measure the difference between two probability distributions for each feature and then find the average value for all the features. From Tables 8.5 and 8.6, all of the class wise synthesized attacks have a low positive KLD value (closer to 0 for most cases) demonstrating a significant similarity to original attacks.

Jensen-Shannon Divergence (JSD)

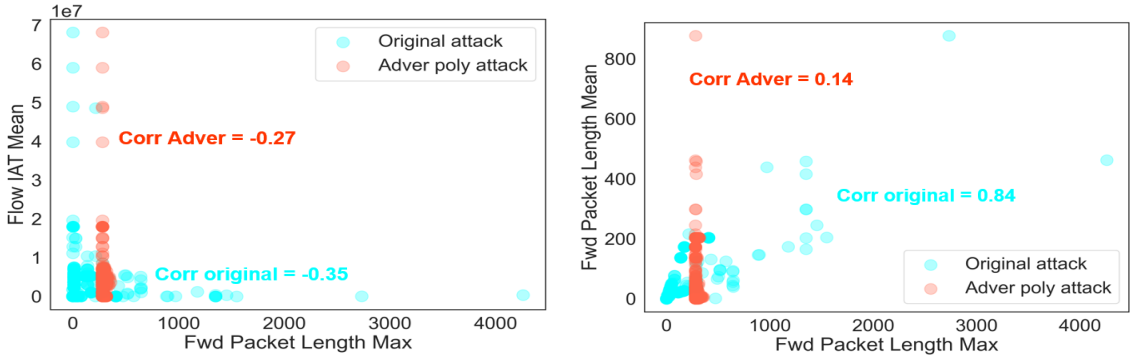
Similar to KLD, we utilize the JSD metric to measure the disparity between two probability distributions for individual features and then subsequently calculate the average value across all the features. Based on the information presented in Tables 8.5 and 8.6, it can be observed that all the synthesized attacks exhibit a low positive JSD value closer to 0. This indicates a substantial similarity to the original attacks.

8.5.3 Correlation analysis

In addition to the distance-based metrics, which quantify the similarity between the two data distributions feature by feature and average these values, we also investigate the interrelationships among the features of the two datasets. To achieve this, we employ a Pearson correlation-based metric to examine the correlation similarity between the two datasets. Initially, we categorize correlation coefficients into six levels as discussed in Table 8.7. Subsequently, we compute the correlation percentage for every feature pair where the original and adversarial synthesized datasets assign similar correlation levels using eq.(6.11) and then calculate the average correlation similarity over the entire dataset.

Table 8.7: Pearson correlation coefficient levels.

Pearson correlation coefficient	Description
Between -1 and -0.5	strong negative
Between -0.5 and -0.3	middle negative
Between -0.3 and -0.1	low negative
Between -0.1 and 0.1	no correlation
Between 0.1 and 0.3	low positive
Between 0.3 and 0.5	middle positive
Between 0.5 and 1	strong positive



(a) *'Fwd Pkt Len Max'* and *'Flow IAT Mean'* (b) *'Fwd Pkt Len Max'* and *'Fwd Pkt Len Mean'*

Figure 8.9: Comparing correlation of feature pairs for a real Slowloris DoS attack and a synthesized adversarial polymorphic attack on CICIDS2017 dataset.

For instance, Fig. 8.9a compares the correlation of features *'Fwd Packet Length Max'* and *'Flow IAT Mean'* for a real Slowloris DoS attack and a synthesized adversarial polymorphic attack. The correlation value for the two features of the real attack is computed as -0.35 whereas for the adversarial polymorphic attack, the value is -0.27. The overall correlation similarity computed using eq. (6.11) is 0.96 which indicates that for the mentioned feature set, the real and adversarial attacks closely resemble each other. Fig. 8.9b compares the correlation of features *'Fwd Packet Length Max'* and *'Fwd Packet Length Mean'* for a real attack and a synthesized attack. The correlation value for the given feature pair for the real attack is 0.84 whereas for the synthesized attack, the value is 0.14. The overall similarity score in this case is calculated as 0.65. Likewise, we compute the correlation similarity for the remaining features in the dataset to determine the overall average.

Tables 8.8 and 8.9 show the average correlation similarity score for the adver-

Table 8.8: Correlation similarity between real attacks and adversarial polymorphic attacks on CICIDS2017 dataset.

Polymorphic attack	Slowloris DoS	Slow Httptest DoS	GoldenEye DoS	Hulk DoS	DDoS
PA1	86.08%	78.79%	82.52%	84.71%	80.60%
PA2	86.04%	81.73%	82.44%	84.29%	80.77%
PA3	85.96%	85.69%	82.42%	84.29%	80.71%

Table 8.9: Correlation similarity between real attacks and adversarial polymorphic attacks on CICIoT2023 dataset.

Polymorphic attack	Brute Force	Web	Recon	Spoof	Mirai	DoS	DDoS
PA1	79.09%	78.33%	82.77%	80.00%	72.23%	81.54%	87.53%
PA2	82.71%	77.81%	80.57%	83.69%	81.11%	84.76%	88.06%
PA3	70.00%	78.04%	82.10%	84.98%	70.71%	88.30%	80.46%

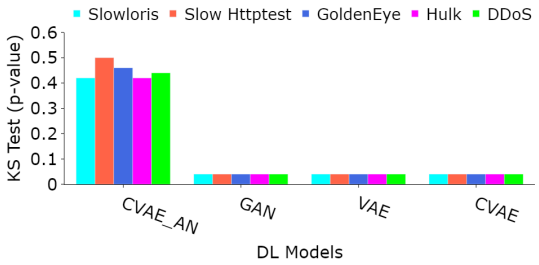
sarially synthesized polymorphic attacks with real attacks on the CICIDS2017 and CICIoT2023 datasets respectively. We observe that the correlation similarity score of adversarial attack data with real attack data is above 70% indicating that the adversarial polymorphic attacks generated by our system have a close semantic resemblance to real attacks.

The overall results using several adversarial attack validation techniques discussed in this section indicate that our system can generate adversarial polymorphic network attacks while maintaining the quality of these attacks.

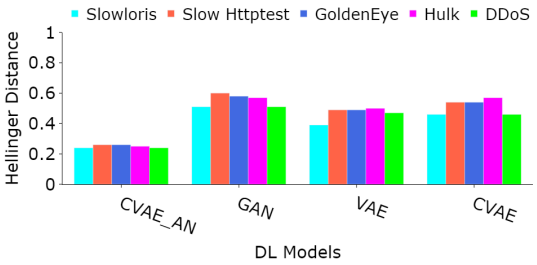
8.5.4 Comparative analysis for synthesized adversarial attack quality

We show the effectiveness of our approach in generating better quality adversarial polymorphic attacks compared to those synthesized by other state-of-the-art DL models such as GAN [254], VAE [255], and CVAE [54]. We select one representative adversarial polymorphic attack from each category for this analysis, but similar results are achieved with other attacks as well.

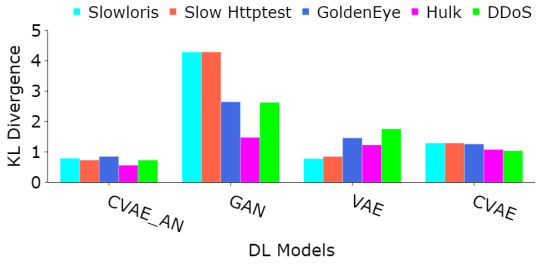
Fig. 8.10 and Fig. 8.11 provide an assessment of the quality of attacks synthesized



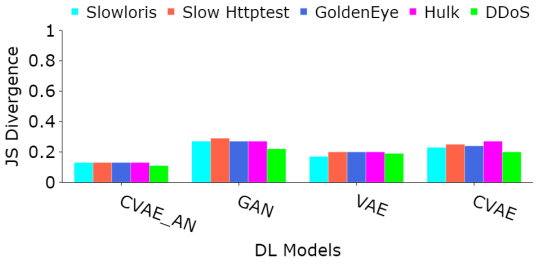
(a) KS Test (p-value)



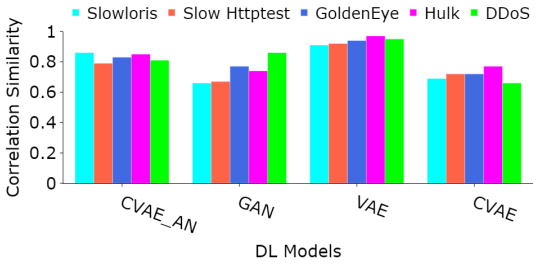
(b) Hellinger distance



(c) KL Divergence



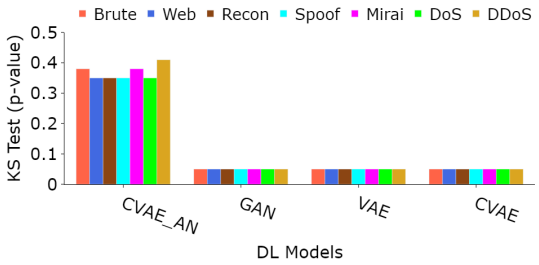
(d) JS Divergence



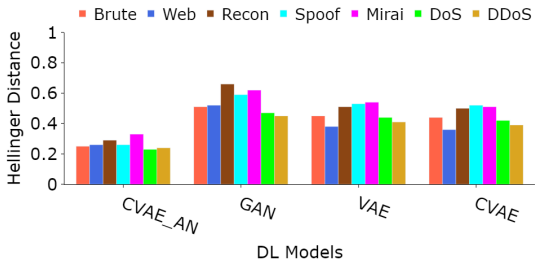
(e) Correlation similarity

Figure 8.10: Comparing the quality of adversarial polymorphic attacks synthesized using multiple state-of-the-art DL models on the CICIDS2017 dataset.

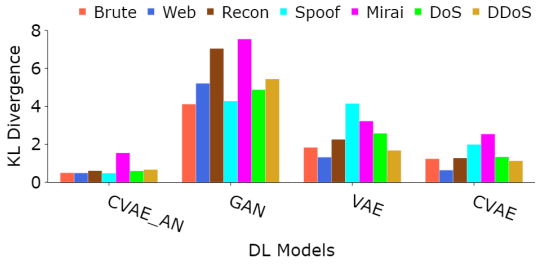
using several generative DL models such as CVAE-AN, GAN, VAE, and CVAE for CICIDS2017 and CICIoT2023 datasets respectively. To corroborate our results, we compare the quality of a synthesized polymorphic attack with an original attack using five tests such as KS test, Hellinger distance, KL divergence (KLD) and JS divergence (JSD), and Correlation similarity. From Fig. 8.10a and Fig. 8.11a, the results for all the cases of the KS test show the p-value for our CVAE-AN model is the highest (greater than or equal to the threshold value of 0.05) indicating that the attacks



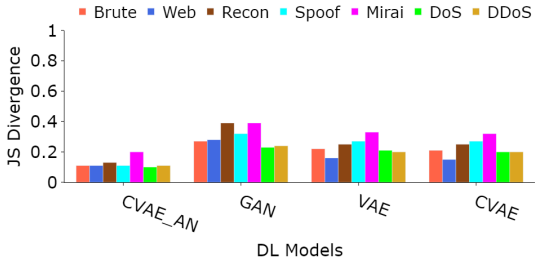
(a) KS Test (p-value)



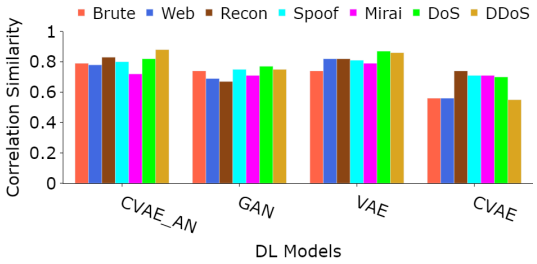
(b) Hellinger distance



(c) KL Divergence



(d) JS Divergence



(e) Correlation similarity

Figure 8.11: Comparing the quality of adversarial polymorphic attacks synthesized using multiple state-of-the-art DL models on the CICIoT2023 dataset.

generated using our model have a similar distribution as that of the original attacks and therefore are of better quality when compared to attacks synthesized using other DL models.

Fig. 8.10b and Fig. 8.11b indicate that the attacks synthesized using CVAE-AN have the lowest distance from real attacks suggesting their close resemblance to original attacks. Additionally, from Fig. 8.10c, 8.11c, 8.10d, and 8.11d, we notice lower values (closer to 0) for KLD and JSD for the adversarial attacks synthesized by

our CVAE-AN model which is indicative of better quality attacks and compared to adversarial attacks synthesized by other DL models.

We observe in Fig. 8.10e and Fig. 8.11e, the correlation similarity of adversarial attacks generated using CVAE-AN to original attacks is consistently higher and more stable than that of the other generative models discussed here. Overall, results suggest the efficacy of the CVAE-AN model in synthesizing realistic adversarial attacks.

For reproducibility, running the experiments under same scenarios, software and hardware environment, hyperparameter settings, dataset versions and sizes, performance metrics, and code versions as described in the thesis should give similar results. However, since CVAE-AN has been generalized using on two datasets, deviations in the results can be expected when it is exposed to other diverse datasets and real-world scenarios.

Chapter 9

Conclusion and Future Directions

9.1 Conclusion

In this thesis, we introduce a novel hybrid model, Conditional Variational Autoencoder-Adversarial Network (CVAE-AN) which combines the best characteristics of CVAE and GAN models. CVAE-AN is employed to adversarially train the polymorphic attacker and the IDS. The attack generator synthesizes realistic dynamically changing attacks for each new cycle. During the polymorphic attack generation phase, the functionality of each attack is preserved by keeping the values of functional features constant. The main objective of the polymorphic attacker is bypassing the detection by the IDS. On the contrary, the main objective of the IDS is to detect the maximum possible polymorphic attacks. Our results demonstrate that during the initial cycles, the polymorphic attacker is successful in eluding detection. However, with incremental retraining, the performance of CVAE-AN IDS improves showing its efficiency against dynamically changing atypical attacks/polymorphic attacks. The comparison of CVAE-AN IDS against other state-of-the-art detection techniques shows improved performance when exposed to polymorphic attacks. Additionally, we apply several

adversarial attack validation techniques such as syntactic validation, hypothesis testing, statistical distance based criteria, and correlation analysis to measure the quality of an attack synthesized by our system. Moreover, we provide a comparative analysis of polymorphic attacks synthesized by multiple state-of-the-art generative DL models. Our empirical findings suggest that CVAE-AN is the best-performing model when synthesizing realistic polymorphic adversarial attacks.

Integration of the attack generator and IDS in CVAE-AN provides a more streamlined, cohesive, and unified approach to cybersecurity. CVAE-AN's ability to continuously adapt to atypical attack patterns is crucial in the face of rapidly changing cyber attacks. While we cannot control when and where an attack is launched, continuous refinement of the NIDS through incremental training allows us to relax the rigid class feature profiles. Beyond acknowledging the potential dangers of polymorphic adversarial attacks, ensuring the quality and functionality of synthesized attacks is crucial to prevent the corruption of IDS training data by invalid adversarial attack samples. The outcomes presented in the thesis contribute to the progress in adversarial machine learning and cybersecurity, improving the resilience and reliability of NIDS against sophisticated polymorphic network threats.

9.2 Advantages and Limitations

The CVAE-AN framework presented in this thesis has demonstrated encouraging outcomes in enhancing the resilience of NIDS against atypical/polymorphic network attacks. Our CVAE-AN attack generator exhibits the capability to produce diverse variations of a single attack by employing conditional generation, adversarial training, and latent space encoding. Through the incorporation of incremental training and data augmentation, the IDS proves adept at adapting to evolving attack strategies.

Leveraging the semisupervised nature of the CVAE-AN model, IDS learns from both labeled and unlabeled data to enhance its ability to identify attacks with varying feature profiles. Furthermore, the assessment of attack quality, involving a range of syntactic and statistical metrics such as KS test, Hellinger distance, KL divergence, JS divergence, and correlation similarity, offers a comprehensive evaluation of the model's effectiveness in synthesizing realistic polymorphic network attacks.

However, it is essential to acknowledge certain limitations. CVAE-AN is a complex model introduced to train both the attack generator and the IDS in adversarial settings. This increased complexity can lead to higher costs during training especially if deployed in real-world scenarios. The sample sizes of training and test datasets may introduce variability in the results. Similar to standard DL models, CVAE-AN may exhibit sensitivity to hyperparameter choices and therefore requires careful tuning to improve the model's effectiveness. The generalization of CVAE-AN results to the datasets employed in this research may be limited. Therefore, the performance of this model in one context may not translate universally. The specific attack scenarios considered in this thesis may not cover the entire spectrum of real-world attacks, potentially leaving certain vulnerabilities unaddressed. Examining the model's performance for only three polymorphic attack cycles per attack class yields limited results. Additionally, while the employed attack quality metrics are necessary to measure the quality of the adversarial synthesized attack, these may not fully capture the model's effectiveness in generating realistic attacks since an in-depth semantic analysis is not considered.

9.3 Future directions

Through this research, we have identified several key aspects for future exploration within the area of DL-based network intrusion detection, with a particular emphasis on addressing atypical and polymorphic network attacks. The primary objective in highlighting these crucial facets is to pave the way for the future development of a well-informed network security solution. The essential aspects for our future investigations are outlined as follows:

9.3.1 Utilizing a wide range of network security datasets

Although our empirical results are based on the analysis of two benchmark datasets named CICIDS2017 and CICIoT2023, our CVAE-AN model is sufficiently generic to be applied to other network datasets. For future work, we further plan to include an evaluation of multiple attack categories, utilizing benchmark datasets such as UNSW-NB15, CSE-CIC-IDS2018, and DDoS 2019. Additionally, we intend to investigate the performance of our IDS against completely unknown network attacks. This involves training the IDS on one network dataset and assessing its performance against another. This can be done to ensure that the IDS is capable of effectively identifying and capturing the characteristics of multiple up-to-date and diversified attacks.

To enhance the applicability of our model across diverse flow-based network security datasets, we aim to identify shared features among these datasets. Our strategy would involve categorizing features into modifiable aspects, including *flow duration*, *total packets/bytes*, *packets/bytes per second*, etc., and non-modifiable aspects such as *port number*, *connection state*, etc. Additionally, we recognize the significance of classifying features based on domain constraints, such as inherent data structure and semantic links, and class constraints, which encompass characteristics distinguishing

one class from another based on functionality.

9.3.2 Online IDS training

CVAE-AN IDS training is done in advance/offline to prepare the IDS for any polymorphic version of a known attack profile, however, for zero-day and unknown attacks the IDS can only keep up by training in real time. This will ensure that the IDS continuously learns from new network data and adjusts to emerging threats, providing a more robust and up-to-date defense mechanism. While our model currently isn't trained in real time, we have plans to extend it for such scenarios in the future. Moreover, to reduce the complexity of the IDS for real networks, we can explore the possibility of developing a lightweight IDS that can satisfactorily model typical, unknown, and atypical/polymorphic network behavior using a less complex and cost-efficient training method.

9.3.3 Semantic validation of flow-based adversarial attacks

The attack quality validation techniques employed in this research are necessary yet may not guarantee better quality adversarial attacks. Therefore, in the future, we would like to focus on identifying precise and explicit network constraints that researchers must adhere to when synthesizing high-quality and realistic adversarial network attacks. For example, one area of focus could be semantic validation by considering the interrelationships among network features.

For an adversarial attack to be feasible and effective, the modifications to correlated features cannot be random. To preserve adversarial attack functionality, any alterations in the feature profile should not diminish the impact of the attack. This can be achieved by eliminating feature dependency during the synthesis of polymor-

phic adversarial attacks. The approach involves categorizing interrelated features into different groups and selecting features from each group to synthesize a polymorphic attack. For example, features such as *Inter Arrival Time (IAT)* and *packets per second* are correlated and should be placed in the same group. When manipulating a feature to synthesize a polymorphic attack, it is crucial to ensure that it does not negatively affect another feature.

Furthermore, in the future, it will be interesting to design a mathematical model capable of explaining feature dependency. Additionally, it is crucial to establish a fixed threshold value to determine an acceptable level of correlation similarity between an original and synthesized attack. The acceptable threshold can further be employed as a benchmark to determine the effectiveness of the adversarial attack generation process.

9.3.4 Packet-based attack manipulation

Current techniques for crafting adversarial network attacks rely on feature manipulation. However, this strategy may prove impractical for intrusion detection systems in real network environments. The challenge lies in the irreversibility of the feature extraction process from raw traffic, unlike in other research areas such as computer vision. The adversarial attacks generated through feature manipulation might struggle to be reintroduced into the network for real-time analysis due to the semantic interdependencies among network features [256]. Consequently, leveraging the feature space for adversarial manipulations in network data may not be a feasible approach. Therefore, our future exploration aims to investigate the feasibility of generating adversarial attacks at the packet level to demonstrate their realism in evading a DL-based IDS.

9.3.5 Comprehensive assessment of generative models for synthesizing realistic attacks

In future research endeavors, it can be compelling to investigate the utilization of alternate generative models such as Conditional GAN (CGAN) [257], Semisupervised GAN (SGAN) [28], Semisupervised Adversarial Autoencoder (SSAAE) [26], among others, to synthesize realistic polymorphic network attacks. Subsequently, a comprehensive assessment of the impact and quality of these synthesized adversarial network attacks on intrusion detection systems can be conducted.

9.3.6 Hyperparameter Optimization

Hyperparameter optimization is a critical step in building an effective DL-based IDS. This process enhances the capability of the system to generalize effectively against sophisticated atypical and polymorphic network attacks and reduces overfitting [2]. Moving forward, we aim to investigate further the requirements of hyperparameter optimization and examine how a single hyperparameter affects the performance and efficiency of the model.

9.3.7 Tracking the evolution of an adversarial polymorphic attack

An adversarial polymorphic network attack poses a significant challenge to the robustness of a DL-based IDS. Consequently, it becomes crucial to analyze and understand the severity of this dynamically changing attack over time. For this reason, in our future work, we plan to focus on analyzing the strength or changes in the intensity of the attack as time progresses with different stages of the attack.

Furthermore, for this research, our attention has been limited to analyzing only 3 polymorphic attack cycles for each attack category. In the future, we aim to examine the performance of both the attacker and IDS over multiple attack cycles within the polymorphic attack chain.

9.3.8 Explainable AI for NIDS

Many Deep Learning-based Network Intrusion Detection Systems (NIDS) function as black boxes, making it challenging to comprehend the rationale behind specific decisions. Explainable AI (XAI) techniques, such as Local Interpretable Model-Agnostic Explanations (LIME) [258] and SHapley Additive exPlanation (SHAP) [241], constitute a rapidly advancing research field that aids in understanding the decision-making process of DL-based IDS. Embracing proper cybersecurity practices becomes imperative for optimizing DL-based NIDS, enhancing detection rates across various attack types (typical, unknown, atypical/polymorphic), and mitigating biased outcomes. A potential avenue for achieving this optimization in the future involves applying XAI to the CVAE-AN IDS model to enhance its interpretability.

9.3.9 Ensemble learning

Ensemble learning combines the detection results of diverse ML/DL models to improve the overall generalization capability. This technique harnesses the strengths of multiple ML/DL models and overcomes their weaknesses [259, 260]. It can be particularly effective in situations such as polymorphic attacks where a single model may struggle due to complexity and rapidly changing attack profiles. Consequently, for our forthcoming research, we intend to investigate further into ensemble learning to enhance the overall detection resilience against atypical/polymorphic attacks.

9.3.10 Human-centricity

DL techniques provide useful insights from input network data and can be employed to automate cyber defense strategies. However, due to the sensitive nature of network security applications and their intolerance towards errors, the real decision-making power lies with the network security expert. Therefore DL-based NIDS needs constant supervision by a security expert, especially in the case of dynamically changing environments where constant updates are needed. Completely replacing cybersecurity experts with a DL model can be possible when it is as creative and intelligent as humans are and can make ethical decisions. This currently seems like a possibility much further in the future.

9.3.11 Non-security applications

Beyond the realm of network security, the CVAE-AN framework, coupled with quality analysis, holds the potential for addressing challenges in various non-security domains. Some potential applications include high quality data generation and detecting anomalies in fields like healthcare, weather forecasting, finance, and drug discovery, where it can assist in generating potential molecular structures. The generative capabilities of CVAE-AN can extend to learning image structures and creating diverse samples for creative arts, as well as transferring styles between images. The introduced quality metrics are valuable for generating more realistic voice/audio samples, benefiting the training of voice recognition systems. Additionally, CVAE-AN can generate diverse, high-quality text samples to augment training data for large language models, contributing to tasks like text completion, summarization, or paraphrasing. The conditional aspect of the model allows precise control over the generated text, for instance, the production of text based on a specific writing style or structure.

Bibliography

[1] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, 2019, pp. 1–6.

[2] U. Sabeel, S. S. Heydari, K. Elgazzar, and K. El-Khatib, "Building an intrusion detection system to detect atypical cyberattack flows," *IEEE Access*, vol. 9, pp. 94 352–94 370, 2021.

[3] U. Sabeel, S. S. Heydari, K. Elgazzar, and K. El-Khatib, "Cvae-an: Atypical attack flow detection using incremental adversarial learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[4] U. Sabeel, S. S. Heydari, K. El-Khatib, and K. Elgazzar, "Unknown, atypical and polymorphic network intrusion detection: A systematic survey," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[5] S. Zavrak and M. İskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108 346–108 358, 2020.

[6] J. Sen and S. Mehtab, "Machine learning applications in misuse and anomaly detection," *Security and privacy from a legal, ethical, and technical perspective*, p. 155, 2020.

[7] M. Usama, M. Asim, S. Latif, J. Qadir, and Ala-Al-Fuqaha, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2019, pp. 78–83.

[8] "Cicflowmeter (formerly iscxflowmeter)," Available at http://www.netflowmeter.ca/netflowmeter.html.

[9] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941

[10] S. Morgan, "Cybercrime infographics:illustrations of the past, present, and future threats we face," Available at https://cybersecurityventures.com/cybercrime-infographic/ (2020/03/24).

[11] R. security solutions, "*Exploring 7 New AI-Powered Cyber Threats, and how RCDevs' Software Can Provide Effective Protection*," Accessed: Jan. 30, 2024. [Online]. Available: https://www.rcdevs.com/.

[12] A. Tarter, "Importance of cyber security," in *Community Policing-A European Perspective*. Springer, 2017, pp. 213–230.

[13] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International*

Conference on Platform Technology and Service (PlatCon). IEEE, 2016, pp. 1–5.

[14] C. Liu, Y. Liu, Y. Yan, and J. Wang, "An intrusion detection model with hierarchical attention mechanism," *IEEE Access*, vol. 8, pp. 67 542–67 554, 2020.

[15] R. A. Shah, Y. Qian, D. Kumar, M. Ali, and M. B. Alvi, "Network intrusion detection through discriminative feature selection by using sparse logistic regression," *Future Internet*, vol. 9, no. 4, p. 81, 2017.

[16] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," in *International Symposium on Computer and Information Sciences*. Springer, 2018, pp. 141–149.

[17] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 228–233.

[18] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary pso and k-nearest neighborhood algorithm," in *2017 11th International Conference on Information & Communication Technology and System (ICTS)*. IEEE, 2017, pp. 181–186.

[19] R. A. Shah, Y. Qian, D. Kumar, M. Ali, and M. B. Alvi, "Network intrusion detection through discriminative feature selection by using sparse logistic regression," *Future Internet*, vol. 9, no. 4, p. 81, 2017.

[20] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature

selection algorithm," in *International Symposium on Computer and Information Sciences*. Springer, 2018, pp. 141–149.

[21] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 228–233.

[22] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, 2019.

[23] J. McHugh, A. Christie, and J. Allen, "Defending yourself: The role of intrusion detection systems," *IEEE Software*, vol. 17, no. 5, pp. 42–51, 2000.

[24] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: supervised or unsupervised?" in *International Conference on Image Analysis and Processing*. Springer, 2005, pp. 50–57.

[25] X. Wang, Y. Du, S. Lin, P. Cui, Y. Shen, and Y. Yang, "advae: A self-adversarial variational autoencoder with gaussian anomaly prior knowledge for anomaly detection," *Knowledge-Based Systems*, vol. 190, p. 105187, 2020.

[26] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[27] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[28] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016.

[29] A. Kumar, P. Sattigeri, and T. Fletcher, "Semi-supervised learning with gans: Manifold invariance with improved inference," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[30] A. Odena, "Semi-supervised learning with generative adversarial networks," *arXiv preprint arXiv:1606.01583*, 2016.

[31] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *2015 National Aerospace and Electronics Conference (NAECON)*. IEEE, 2015, pp. 339–344.

[32] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.

[33] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48 231–48 246, 2018.

[34] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 2003 SIAM international conference on data mining*. SIAM, 2003, pp. 25–36.

[35] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[37] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[38] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sensors Letters*, vol. 3, no. 1, pp. 1–4, 2019.

[39] J.-H. Seo and Y.-H. Kim, "Machine-learning approach to optimize smote ratio in class imbalance dataset for intrusion detection," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[40] X. Tan, S. Su, Z. Huang, X. Guo, Z. Zuo, X. Sun, and L. Li, "Wireless sensor networks intrusion detection based on smote and the random forest algorithm," *Sensors*, vol. 19, no. 1, p. 203, 2019.

[41] H. Zhang, L. Huang, C. Q. Wu, and Z. Li, "An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset," *Computer Networks*, vol. 177, p. 107315, 2020.

[42] Z. Hu, L. Wang, L. Qi, Y. Li, and W. Yang, "A novel wireless network intrusion detection method based on adaptive synthetic sampling and an improved convolutional neural network," *IEEE Access*, vol. 8, pp. 195 741–195 751, 2020.

[43] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm," *Computers & Security*, p. 102289, 2021.

[44] S. Huang and K. Lei, "Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Networks*, vol. 105, p. 102177, 2020.

[45] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skorić, "Measuring intrusion detection capability: an information-theoretic approach," in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006, pp. 90–101.

[46] B. Abma, "Evaluation of requirements management tools with support for traceability-based change impact analysis," *M.S. thesis, Dept. Elect. Eng., Univ. Twente, Enschede, The Netherlands*, 2009.

[47] V. R. Ranganath. Informedness and markedness. Accessed: Apr. 10, 2021. [Online]. Available: https://rvprasad.medium.com/informedness-and-markedness-20e3f54d63bc

[48] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[49] A. S. Qureshi, A. Khan, N. Shamim, and M. H. Durad, "Intrusion detection using deep sparse auto-encoder and self-taught learning," *Neural Computing and Applications*, pp. 1–13, 2019.

[50] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020, pp. 37–45.

[51] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, "Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset," *IEEE Access*, vol. 9, pp. 140 136–140 146, 2021.

[52] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[53] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 5597–5621, 2019.

[54] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[55] A. Hannan, C. Gruhl, and B. Sick, "Anomaly based resilient network intrusion detection using inferential autoencoders," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 1–7.

[56] E. P. Jiang, "Automatic text classification from labeled and unlabeled data," in *Intelligent Data Analysis for Real-Life Applications: Theory and Practice*. IGI Global, 2012, pp. 249–264.

[57] Y. Wu, L. Nie, S. Wang, Z. Ning, and S. Li, "Intelligent intrusion detection for internet of things security: A deep convolutional generative adversarial network-enabled approach," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[58] A. Liu, Y. Wang, and T. Li, "Sfe-gacn: A novel unknown attack detection under insufficient data via intra categories generation in embedding space," *Computers & Security*, vol. 105, p. 102262, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821000869

[59] J. Zhou, Z. Wu, Y. Xue, M. Li, and D. Zhou, "Network unknown-threat detection based on a generative adversarial network and evolutionary algorithm," *International Journal of Intelligent Systems*, vol. 37, no. 7, pp. 4307–4328, 2022.

[60] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Variational data generative model for intrusion detection," *Knowledge and Information Systems*, vol. 60, no. 1, pp. 569–590, 2019.

[61] L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Learning latent distribution for distinguishing network traffic in intrusion detection system," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[62] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42 169–42 184, 2020.

[63] K. Hara and K. Shiomoto, "Intrusion detection system using semi-supervised learning with adversarial auto-encoder," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–8.

[64] F. Aloul, I. Zualkernan, N. Abdalgawad, L. Hussain, and D. Sakhnini, "Network intrusion detection on the iot edge using adversarial autoencoders," in *2021 International Conference on Information Technology (ICIT)*, 2021, pp. 120–125.

[65] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul, "Generative deep learning to detect cyberattacks for the iot-23 dataset," *IEEE Access*, pp. 1–1, 2021.

[66] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, 2019.

[67] S. Azmin and A. M. A. A. Islam, "Network intrusion detection system based on conditional variational laplace autoencoder," in *7th International Conference on Networking, Systems and Security*, 2020, pp. 82–88.

[68] X. Xu, J. Li, Y. Yang, and F. Shen, "Toward effective intrusion detection using log-cosh conditional variational autoencoder," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6187–6196, 2021.

[69] J. Vitorino, N. Oliveira, and I. Praça, "Adaptative perturbation patterns: realistic adversarial learning for robust intrusion detection," *Future Internet*, vol. 14, no. 4, p. 108, 2022.

[70] U. Sabeel, S. S. Heydari, K. El-Khatib, and K. Elgazzar, "Incremental adversarial learning for polymorphic attack detection," manuscript submitted to IEEE Transactions on Machine Learning in Communications and Networking, 2023.

[71] R. Chauhan, U. Sabeel, A. Izaddoost, and S. Shah Heydari, "Polymorphic adversarial cyberattacks using wgan," *Journal of Cybersecurity and Privacy*, vol. 1, no. 4, pp. 767–792, 2021.

[72] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2006, pp. 81–105.

[73] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," *arXiv preprint arXiv:1809.02077*, 2018.

[74] S. Mahdavifar and A. A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, 2019.

[75] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[76] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[77] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of ids for iot: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, 2021.

[78] "Neglected machine learning ideas," Available at https://scottlocklin.wordpress.com/2014/07/22/neglected-machine-learning-ideas/ (2014/07/22).

[79] C. Ford, "Understanding q-q plots — university of virginia library research data services + sciences." Available at https://data.library.virginia.edu/understanding-q-q-plots/ (2015).

[80] "Principal component analysis (pca)," Available at https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[81] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[82] S. Glen, "*Correlation Coefficient: Simple Definition, Formula, Easy Steps*," Accessed: Nov. 11, 2021. [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/, StatisticsHowTo.com: Elementary Statistics for the rest of us!

[83] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

[84] H. O. Lancaster and E. Seneta, "Chi-square distribution," *Encyclopedia of biostatistics*, vol. 2, 2005.

[85] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," *Institute for Signal and information Processing*, vol. 18, no. 1998, pp. 1–8, 1998.

[86] I. Karna, A. Madam, C. Deokule, R. Adhao, and V. Pachghare, "Ensemble-based filter feature selection technique for building flow-based ids," in *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, 2021, pp. 324–328.

[87] A. Attia, M. Faezipour, and A. Abuzneid, "Network intrusion detection with xgboost and deep learning algorithms: An evaluation study," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2020, pp. 138–143.

[88] S. Shakeela, N. S. Shankar, P. M. Reddy, T. K. Tulasi, and M. M. Sai, "Optimal ensemble learning based on distinctive feature selection by univariate anova-f statistics for ids," *International Journal of Electronics and Telecommunications*, vol. 67, no. 2, pp. 267–275, 2021.

[89] Y. Liu, Z. Xu, J. Yang, L. Wang, C. Song, and K. Chen, "A novel meta-heuristic-based sequential forward feature selection approach for anomaly detection sys-

tems," in *2016 International Conference on Network and Information Systems for Computers (ICNISC)*, 2016, pp. 218–227.

[90] O. Al-Jarrah, A. Siddiqui, M. Elsalamouny, P. Yoo, S. Muhaidat, and K. Kim, "Machine-learning-based feature selection techniques for large-scale network intrusion detection," in *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2014, pp. 177–181.

[91] S. Ustebay, Z. Turgut, and M. A. Aydin, "Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 2018, pp. 71–76.

[92] J. Ranstam and J. Cook, "Lasso regression," *Journal of British Surgery*, vol. 105, no. 10, pp. 1348–1348, 2018.

[93] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, no. 1, pp. 3–20, 1975.

[94] Y. Hua, "An efficient traffic classification scheme using embedded feature selection and lightgbm," in *2020 Information Communication Technologies Conference (ICTC)*, 2020, pp. 125–130.

[95] T.-T. Nguyen, J. Z. Huang, and T. T. Nguyen, "Unbiased feature selection in learning random forests for high-dimensional data," *The Scientific World Journal*, vol. 2015, 2015.

[96] "Features — lightgbm 2.3.2 documentation." Available at https://lightgbm.readthedocs.io/en/latest/Features.html.

[97] L. Van Der Maaten, E. Postma, J. Van den Herik *et al.*, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[98] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, pp. 195–200.

[99] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Computers & Security*, p. 101851, 2020.

[100] S. Seo, S. Park, and J. Kim, "Improvement of network intrusion detection accuracy by using restricted boltzmann machine," in *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2016, pp. 413–417.

[101] P. Kline, *An easy guide to factor analysis*. Routledge, 2014.

[102] F. Meng, Y. Fu, F. Lou, and Z. Chen, "An effective network attack detection method based on kernel pca and lstm-rnn," in *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*, 2017, pp. 568–572.

[103] R. Zhao, G. Gui, Z. Xue, J. Yin, T. Ohtsuki, B. Adebisi, and H. Gacanin, "A novel intrusion detection method based on lightweight neural network for internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[104] N. Wu and J. Zhang, "Factor analysis based anomaly detection," in *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003.*, 2003, pp. 108–115.

[105] scikit learn, "*Manifold learning*," Accessed: Nov. 11, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/manifold.html#locally-linear-embedding.

[106] K.-m. Zheng, X. Qian, and P.-c. Wang, "Dimension reduction in intrusion detection using manifold learning," in *2009 International Conference on Computational Intelligence and Security*, vol. 2, 2009, pp. 464–468.

[107] G. Hou, X. Ma, and Y. Zhang, "A new method for intrusion detection using manifold learning algorithm," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 12, pp. 7339–7343, 2013.

[108] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[109] Y. Hamid and M. Sugumaran, "A t-sne based non linear dimension reduction for network intrusion detection," *International Journal of Information Technology*, vol. 12, no. 1, pp. 125–134, 2020.

[110] H. Yao, C. Li, and P. Sun, "Using parametric t-distributed stochastic neighbor embedding combined with hierarchical neural network for network intrusion detection." *Int. J. Netw. Secur.*, vol. 22, no. 2, pp. 265–274, 2020.

[111] G. B. Souza, D. F. S. Santos, R. G. Pires, A. N. Marana, and J. P. Papa, "A restricted boltzmann machine-based approach for robust dimensionality reduction," in *2017 Workshop of Computer Vision (WVC)*, 2017, pp. 138–143.

[112] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[113] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[114] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, 2017, pp. 63–69.

[115] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "Intrusion detection in smart cities using restricted boltzmann machines," *Journal of Network and Computer Applications*, vol. 135, pp. 76–83, 2019.

[116] G. Zhao, C. Zhang, and L. Zheng, "Intrusion detection using deep belief network and probabilistic neural network," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1, 2017, pp. 639–642.

[117] R. Elshawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," *arXiv preprint arXiv:1906.02287*, 2019.

[118] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated machine learning*. Springer, Cham, 2019, pp. 3–33.

[119] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 481–488.

[120] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[121] A. Dey, "Deep ids : A deep learning approach for intrusion detection based on ids 2018," in *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 2020, pp. 1–5.

[122] Y. Li, A. Dandoush, and J. Liu, "Evaluation and optimization of learning-based dns over https traffic classification," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–6.

[123] S. Zheng, "Network intrusion detection model based on convolutional neural network," in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5, 2021, pp. 634–637.

[124] S. A. Althubiti, E. M. Jones, and K. Roy, "Lstm for anomaly-based network intrusion detection," in *2018 28th International telecommunication networks and applications conference (ITNAC)*. IEEE, 2018, pp. 1–3.

[125] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing," *ICT Express*, vol. 5, no. 3, pp. 211–214, 2019.

[126] M. Choraś and M. Pawlicki, "Intrusion detection approach based on optimised artificial neural network," *Neurocomputing*, vol. 452, pp. 705–715, 2021.

[127] L. Ashiku and C. Dagli, "Network intrusion detection system using deep learning," *Procedia Computer Science*, vol. 185, pp. 239–247, 2021.

[128] G. Zizzo, C. Hankin, S. Maffeis, and K. Jones, "Adversarial attacks on time-series intrusion detection for industrial control systems," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 899–910.

[129] N. Lee, S. Y. Ooi, and Y. H. Pang, "A sequential approach to network intrusion detection," in *Computational Science and Technology*. Springer, 2020, pp. 11–21.

[130] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, 2019.

[131] M. Labonne, A. Olivereau, B. Polvé, and D. Zeghlache, "A cascade-structured meta-specialists approach for neural network-based intrusion detection," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–6.

[132] 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 of Things Journal*, pp. 1–1, 2021.

[133] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *Journal of Information Security and Applications*, vol. 58, p. 102804, 2021.

[134] S. Naseer and Y. Saleem, "Enhanced network intrusion detection using deep convolutional neural networks," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 12, no. 10, pp. 5159–5178, 2018.

[135] M. A. Khan, M. Karim, Y. Kim *et al.*, "A scalable and hybrid intrusion detection system based on the convolutional-lstm network," *Symmetry*, vol. 11, no. 4, p. 583, 2019.

[136] M. A. Khan, "Hcrnnids: Hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, p. 834, 2021.

[137] M. S. Ansari, V. Bartoš, and B. Lee, "Gru-based deep learning approach for network intrusion alert prediction," *Future Generation Computer Systems*, vol. 128, pp. 235–247, 2022.

[138] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 104–119.

[139] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[140] Y. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Improving classification attacks in iot intrusion detection system using bayesian hyperparameter optimization," in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, 2020, pp. 146–151.

[141] V. Awatramani and P. Gupta, "Intrusion detection using nature-inspired algorithms and automated machine learning," *Smart and Sustainable Intelligent Systems*, pp. 295–306, 2021.

[142] D. Schubert, H. Eikerling, and J. Holtmann, "Application-aware intrusion detection: A systematic literature review, implications for automotive systems, and applicability of automl," *Frontiers in Computer Science*, vol. 3, p. 75, 2021.

[143] F. Jamil, D. Kim *et al.*, "An ensemble of a prediction and learning mechanism for improving accuracy of anomaly detection in network intrusion environments," *Sustainability*, vol. 13, no. 18, p. 10057, 2021.

[144] Y. Gormez, Z. Aydin, R. Karademir, and V. C. Gungor, "A deep learning approach with bayesian optimization and ensemble classifiers for detecting denial of service attacks," *International Journal of Communication Systems*, vol. 33, no. 11, p. e4401, 2020.

[145] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[146] W. Elmasry, A. Akbulut, and A. H. Zaim, "Empirical study on multiclass classification-based network intrusion detection," *Computational Intelligence*, vol. 35, no. 4, pp. 919–954, 2019.

[147] Y. Chen, S. Chen, M. Xuan, Q. Lin, and W. Wei, "Evolutionary convolutional neural network: An application to intrusion detection," in *2021 13th International Conference on Advanced Computational Intelligence (ICACI)*, 2021, pp. 245–252.

[148] I. V. Pustokhina, D. A. Pustokhin, E. L. Lydia, P. Garg, A. Kadian, and K. Shankar, "Hyperparameter search based convolution neural network with bi-lstm model for intrusion detection system in multimedia big data environment," *Multimedia Tools and Applications*, pp. 1–18, 2021.

[149] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[150] A. Talos, "*Hyperparameter Optimization for Keras, TensorFlow (tf.keras) and PyTorch*," Accessed: Nov. 17, 2021. [Online]. Available: https://github.com/autonomio/talos.

[151] J. Zhou, A. Velichkevich, K. Prosvirov, A. Garg, Y. Oshima, and D. Dutta, "Katib: A distributed general automl platform on kubernetes," in *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)*, 2019, pp. 55–57.

[152] E. LeDell and S. Poirier, "H2o automl: Scalable automatic machine learning," in *Proceedings of the AutoML Workshop at ICML*, vol. 2020, 2020.

[153] D. AI, "*Determined: Deep Learning Training Platform*," Accessed: Nov. 17, 2021. [Online]. Available: https://github.com/determined-ai/determined.

[154] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[155] J. Bergstra, D. Yamins, D. D. Cox *et al.*, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*, vol. 13. Citeseer, 2013, p. 20.

[156] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, vol. 28, 2015, pp. 2962–2970.

[157] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[158] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523.

[159] E. Bakshy, M. Balandat, and K. Kashin, "*Open-sourcing Ax and botorch: New AI tools for Adaptive Experimentation*," Accessed: Nov. 17, 2021. [Online]. Available: https://ai.facebook.com/blog/open-sourcing-ax-and-botorch-new-ai-tools-for-adaptive-experimentation/.

[160] P. Probst, M. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *ArXiv preprint arXiv:1804.03515*, 2018. [Online]. Available: https://arxiv.org/abs/1804.03515

[161] J. Davison, "*DEvol - Deep Neural Network Evolution*," Accessed: Nov. 17, 2021. [Online]. Available: https://github.com/joeddav/devol.

[162] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[163] J. Rapin and O. Teytaud, "*Nevergrad - A gradient-free optimization platform*," Accessed: Nov. 17, 2021. [Online]. Available: https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[164] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in *Proceedings of the genetic and evolutionary computation conference 2016*, 2016, pp. 485–492.

[165] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, "Automating biomedical data science through tree-based pipeline

optimization," in *European conference on the applications of evolutionary computation*. Springer, 2016, pp. 123–137.

[166] M. Kianpour and S.-F. Wen, "Timing attacks on machine learning: State of the art," in *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 1*. Springer, 2020, pp. 111–125.

[167] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2154–2156.

[168] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[169] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[170] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[171] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, pp. 3483–3491, 2015.

[172] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.

[173] N. Khare, P. Devan, C. L. Chowdhary, S. Bhattacharya, G. Singh, S. Singh, and B. Yoon, "Smo-dnn: Spider monkey optimization and deep neural network hybrid classifier model for intrusion detection," *Electronics*, vol. 9, no. 4, p. 692, 2020.

[174] P. Bedi, N. Gupta, and V. Jindal, "I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems," *Applied Intelligence*, vol. 51, no. 2, pp. 1133–1151, 2021.

[175] M. Lei, X. Li, B. Cai, Y. Li, L. Liu, and W. Kong, "P-dnn: An effective intrusion detection method based on pruning deep neural network," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[176] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang, "Pccn: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows," *IEEE Access*, vol. 7, pp. 119 904–119 916, 2019.

[177] S. Yang, M. Tan, S. Xia, and F. Liu, "A method of intrusion detection based on attention-lstm neural network," in *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, 2020, pp. 46–50.

[178] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional lstm deep learning approach for intrusion detection," *Expert Systems with Applications*, vol. 185, p. 115524, 2021.

[179] P. R. Kanna and P. Santhi, "Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features," *Knowledge-Based Systems*, vol. 226, p. 107132, 2021.

[180] M. Manavi and Y. Zhang, "A new intrusion detection system based on gated recurrent unit (gru) and genetic algorithm," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2019, pp. 368–383.

[181] M. V. Assis, L. F. Carvalho, J. Lloret, and M. L. Proença Jr, "A gru deep learning system against attacks in software defined networks," *Journal of Network and Computer Applications*, vol. 177, p. 102942, 2021.

[182] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3540–3552, 2020.

[183] S. Ho, S. A. Jufout, K. Dajani, and M. Mozumdar, "A novel intrusion detection model for detecting known and innovative cyberattacks using convolutional neural network," *IEEE Open Journal of the Computer Society*, vol. 2, pp. 14–25, 2021.

[184] W. H. Bangyal, J. Ahmad, H. T. Rauf, and R. Shakir, "Evolving artificial neural networks using opposition based particle swarm optimization neural network for data classification," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. IEEE, 2018, pp. 1–6.

[185] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in *2018 10th international conference on cyber Conflict (CyCon)*. IEEE, 2018, pp. 371–390.

[186] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for iot based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31 711–31 722, 2019.

[187] S. Manimurugan, S. Al-Mutairi, M. M. Aborokbah, N. Chilamkurti, S. Ganesan, and R. Patan, "Effective attack detection in internet of medical things smart environment using a deep belief neural network," *IEEE Access*, vol. 8, pp. 77 396–77 404, 2020.

[188] Z. Wang, Y. Zeng, Y. Liu, and D. Li, "Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection," *IEEE Access*, vol. 9, pp. 16 062–16 091, 2021.

[189] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 91–99.

[190] J. Yang, X. Chen, S. Chen, X. Jiang, and X. Tan, "Conditional variational auto-encoder and extreme value theory aided two-stage learning approach for intelligent fine-grained known/unknown intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3538–3553, 2021.

[191] X. Mao, Q. Li, H. Xie, R. Y. Lau, and Z. Wang, "Multi-class generative adversarial networks with the l2 loss function," *arXiv preprint arXiv:1611.04076*, vol. 5, pp. 1057–7149, 2016.

[192] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13 546–13 560, 2019.

[193] Z. Zhang, Y. Zhang, D. Guo, and M. Song, "A scalable network intrusion detection system towards detecting, discovering, and learning unknown attacks," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 6, pp. 1649–1665, 2021.

[194] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "A deep learning ensemble approach to detecting unknown network attacks," *Journal of Information Security and Applications*, vol. 67, p. 103196, 2022.

[195] C.-S. Shieh, W.-W. Lin, T.-T. Nguyen, C.-H. Chen, M.-F. Horng, and D. Miu, "Detection of unknown ddos attacks with deep learning and gaussian mixture model," *Applied Sciences*, vol. 11, no. 11, p. 5213, 2021.

[196] H. Du, S. Wang, and H. Huo, "Xfinder: detecting unknown anomalies in distributed machine learning scenario," *Front. Comput. Sci. 3: 710384. doi: 10.3389/fcomp*, 2021.

[197] X. Hu, C. Gu, Y. Chen, X. Chen, and F. Wei, "Opencbd: A network-encrypted unknown traffic identification scheme based on open-set recognition," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[198] C.-S. Shieh, T.-T. Nguyen, C.-Y. Chen, and M.-F. Horng, "Detection of unknown ddos attack using reconstruct error and one-class svm featuring stochastic gradient descent," *Mathematics*, vol. 11, no. 1, p. 108, 2022.

[199] R. Chauhan and S. Shah Heydari, "Polymorphic adversarial ddos attack on ids using gan," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–6.

[200] H. S. Vu, D. Ueta, K. Hashimoto, K. Maeno, S. Pranata, and S. M. Shen, "Anomaly detection with adversarial dual autoencoders," *arXiv preprint arXiv:1902.06924*, 2019.

[201] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[202] D. Terzopoulos *et al.*, "Multi-adversarial variational autoencoder networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 777–782.

[203] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Cvae-gan: fine-grained image generation through asymmetric training," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2745–2754.

[204] H. A. Alatwi and C. Morisset, "Adversarial machine learning in network intrusion detection domain: A systematic review," *arXiv preprint arXiv:2112.03315*, 2021.

[205] M. A. Merzouk, F. Cuppens, N. Boulahia-Cuppens, and R. Yaich, "Investigating the practicality of adversarial evasion attacks on network intrusion detection," *Annals of Telecommunications*, pp. 1–13, 2022.

[206] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[207] T. George Karimpanal and R. Bouffanais, "Self-organizing maps for storage and transfer of knowledge in reinforcement learning," *Adaptive Behavior*, vol. 27, no. 2, pp. 111–126, 2019.

[208] J. Zhao, S. Shetty, and J. W. Pan, "Feature-based transfer learning for network security," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, 2017, pp. 17–22.

[209] J. Zhao, S. Shetty, J. W. Pan, C. Kamhoua, and K. Kwiat, "Transfer learning for detecting unknown network attacks," *EURASIP Journal on Information Security*, vol. 2019, no. 1, pp. 1–13, 2019.

[210] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Deep transfer learning for iot attack detection," *IEEE Access*, vol. 8, pp. 107 335–107 344, 2020.

[211] J. He, Y. Tan, W. Guo, and M. Xian, "A small sample ddos attack detection method based on deep transfer learning," in *2020 International Conference on Computer Communication and Network Security (CCNS)*, 2020, pp. 47–50.

[212] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2015.

[213] N. Joshi, " *Exploring the limits of transfer learning*," Accessed: Feb. 15, 2022. [Online]. Available: https://www.allerin.com/blog/exploring-the-limits-of-transfer-learning.

[214] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *arXiv preprint arXiv:1811.12560*, 2018.

[215] MIT, "*Introduction to Deep Learning*," Accessed: July. 15, 2021. [Online]. Available: http://introtodeeplearning.com/.

[216] C. Kim and J. Park, "Designing online network intrusion detection using deep auto-encoder q-learning," *Computers & Electrical Engineering*, vol. 79, p. 106460, 2019.

[217] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. 112963, 2020.

[218] T. V. Phan, S. Sultana, T. G. Nguyen, and T. Bauschert, "¡tex¿q¡/tex¿ - transfer: A novel framework for efficient deep transfer learning in networking," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2020, pp. 146–151.

[219] A. Venturi, G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, "Drelab-deep reinforcement learning adversarial botnet: A benchmark dataset for adversarial attacks against botnet intrusion detection systems," *Data in Brief*, vol. 34, p. 106631, 2021.

[220] K. Sethi, Y. V. Madhav, R. Kumar, and P. Bera, "Attention based multi-agent intrusion detection systems using reinforcement learning," *Journal of Information Security and Applications*, vol. 61, p. 102923, 2021.

[221] Z. Li, Z. Qin, and P. Shen, "Intrusion detection via wide and deep model," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 717–730.

[222] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[223] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[224] M. A. Merzouk, F. Cuppens, N. Boulahia-Cuppens, and R. Yaich, "A deeper analysis of adversarial examples in intrusion detection," in *Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, November 4–6, 2020, Revised Selected Papers 15*. Springer, 2021, pp. 67–84.

[225] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[226] D. Civin, "Explainable ai could reduce the impact of biased algorithms — venturebeat." Available at https://venturebeat.com/2018/05/21/explainable-ai-could-reduce-the-impact-of-biased-algorithms/.

[227] G. Brown, "Ensemble learning." *Encyclopedia of Machine Learning*, vol. 312, 2010.

[228] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Information Fusion*, vol. 52, pp. 1–12, 2019.

[229] Codeacademy, "Normalization," Available at https://www.codecademy.com/articles/normalization.

[230] A. Bhandari, "Feature scaling for machine learning," Available at https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.

[231] "Preprocessing data," Available at https://scikit-learn.org/stable/modules/preprocessing.html.

[232] "Slowhttptest package description," Available at https://tools.kali.org/stress-testing/slowhttptest.

[233] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. SCITEPRESS, 2018, pp. 108–116.

[234] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *Proceedings of the 3rd International Conference on Information System Security and Privacy (ICISSP)*. SCITEPRESS, 2017, pp. 253–262.

[235] "Decision tree classifier," Available at https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

[236] E. L. at UPenn, "A python automated machine learning tool that optimizes machine learning pipelines using genetic programming," Available at https://github.com/EpistasisLab/tpot.

[237] N. Bakhareva, A. Shukhman, A. Matveev, P. Polezhaev, Y. Ushakov, and L. Legashev, "Attack detection in enterprise networks by machine learning methods," in *2019 International Russian Automation Conference (RusAutoCon)*. IEEE, 2019, pp. 1–6.

[238] M. Azizjon, A. Jumabek, and W. Kim, "1d cnn based network intrusion detection with normalization on imbalanced data," in *2020 International Conference*

on Artificial Intelligence in Information and Communication (ICAIIC), 2020, pp. 218–224.

[239] M. D. Hossain, H. Ochiai, D. Fall, and Y. Kadobayashi, "Lstm-based network attack detection: Performance comparison by hyper-parameter values tuning," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2020, pp. 62–69.

[240] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[241] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

[242] C.-C. Ma, B.-Y. Wu, Y.-B. Fan, Y. Zhang, and Z.-F. Li, "Effective and robust detection of adversarial examples via benford-fourier coefficients," *Machine Intelligence Research*, pp. 1–17, 2022.

[243] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[244] D. D. Boos and L. A. Stefanski, "P-value precision and reproducibility," *The American Statistician*, vol. 65, no. 4, pp. 213–221, 2011.

[245] E. Hellinger, "New foundation of the theory of quadratic forms of infinitely many variable ones." *Journal of pure and applied mathematics*, vol. 1909, no. 136, pp. 210–271, 1909.

[246] Y. Bu, S. Zou, Y. Liang, and V. V. Veeravalli, "Estimation of kl divergence: Optimal minimax rate," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2648–2674, 2018.

[247] M. Menéndez, J. Pardo, L. Pardo, and M. Pardo, "The jensen-shannon divergence," *Journal of the Franklin Institute*, vol. 334, no. 2, pp. 307–318, 1997.

[248] *Synthetic Data Metrics*, DataCebo, Inc., 10 2022, version 0.8.0. [Online]. Available: https://docs.sdv.dev/sdmetrics/

[249] "Simple slowloris in python," Available at https://github.com/gkbrk/slowloris.

[250] "Goldeneye," Available at https://github.com/jseidl/GoldenEye.

[251] "Hulk dos tool," Available at https://github.com/grafov/hulk.

[252] "High orbit ion cannon," Available at https://sourceforge.net/projects/highorbitioncannon/.

[253] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[254] S. Sapre, K. Islam, and P. Ahmadi, "A comprehensive data sampling analysis applied to the classification of rare iot network intrusion types," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–2.

[255] C. Liu, R. Antypenko, I. Sushko, and O. Zakharchenko, "Intrusion detection system after data augmentation schemes based on the vae and cvae," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 1000–1010, 2022.

[256] I. Debicha, B. Cochez, T. Kenaza, T. Debatty, J.-M. Dricot, and W. Mees, "Adv-bot: Realistic adversarial botnet attacks against network intrusion detection systems," *Computers & Security*, vol. 129, p. 103176, 2023.

[257] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[258] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 1135–1144.

[259] Y. Alotaibi and M. Ilyas, "Ensemble-learning framework for intrusion detection to enhance internet of things devices security," *Sensors*, vol. 23, no. 12, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/12/5568

[260] N. Thockchom, M. M. Singh, and U. Nandi, "A novel ensemble learning-based model for network intrusion detection," *Complex & Intelligent Systems*, pp. 1–22, 2023.

Appendix A

CICIDS2017/CICIoT2023 dataset

A detailed description of the CICIDS2017 dataset including the number of flows, percentages of attacks, attack type, and their description is given in table A.1. For our research, we have used five classes of attack and one normal class. All the attacks belong to the Denial of Service (DoS) or Distributed Denial of Service (DDoS) categories. The table explains in detail each attack category and its description. As seen in the table, the DoS Hulk attack and DDoS attack classes are volumetric attacks that overwhelm the victim server with a huge number of incoming packets. On the other hand, GoldenEye, Slowloris, and Slowhttptest DoS attacks exploit the HTTP vulnerability by leaving a huge number of half open connections and exhausting all the victim server connections, thus denying access to other legitimate users. Table A.2 and A.3 describe all the features in the CICIDS2017 dataset.

A comprehensive overview of the CICIoT2023 dataset is presented in Table A.4. This table provides the details such as flow count, percentage of each attack class and normal data, and types of attacks with their description. Table A.5 describes all the features in the CICIoT2023 dataset. Table A.6 and Table A.7 present a compilation of acronyms utilized throughout this study.

Table A.1: CICIDS2017 Dataset Class Description

Class	**Total Flows**	**Percentage**	**Attack Type**	**Description**
Normal	667887	63.70%	Non malicious data	This represents data from normal user activities.
DoS Hulk	231073	22.04%	Volumetric	A large number of malicious obscure HTTP requests are sent to the victim server by the attacker to exhaust all its resources and render it useless for legitimate clients. Also known as HTTP Unbearable Load King.
DDoS	128027	12.21%	Volumetric	The attacker controls other hosts (bots) that overwhelm a victim server by sending multiple HTTP/TCP/UDP packets. The victim server cannot handle excessive load from the attacker hosts, thereby making the resources unavailable for legitimate clients.
DoS GoldenEye	10293	0.98%	HTTP vulnerability exploit	An Application layer based DoS attack where the attacker keeps the connection with the victim HTTP server alive until it consumes all the sockets on the victim. The victim server is rendered useless for legitimate requests.
DoS Slowloris	5796	0.55%	HTTP vulnerability exploit	Slow HTTP header attack where partial HTTP GET/-POST requests are sent to the victim server to keep as many open connections as possible. The victim server gets overwhelmed and is unable to cater to legitimate connection requests.
DoS Slowhttptest	5499	0.52%	HTTP vulnerability exploit	Just like Slowloris attack, this attack too keeps the victim HTTP server resources engaged while waiting for incomplete requests to finish. Due to the long wait times, legitimate clients are denied service since all connections are busy. Slow Httptest attack can be launched in different modes such as slow header, slow body, and slow read [232].
Total	1048575	100%	-	-

Table A.2: List of CICIDS2017 Features [8]

Feature	Description
flow duration	Duration of flow in microseconds.
tot fwd pkts	Total packets transmitted from sender to receiver (forward direction)
tot bwd pkts	Total packets transmitted from receiver to sender (backward direction)
tot len fwd pkt	Total size of packets in forward direction
tot len bwd pkt	Total size of packets in backward direction
fwd pkt len min	Minimum size of packet in forward direction
fwd pkt len max	Maximum size of packet in forward direction
fwd pkt len mean	Mean size of the packet in forward direction
fwd pkt len std	Standard deviation size of the packet in forward direction
bwd pkt len min	Minimum size of packet in backward direction
bwd pkt len max	Maximum size of the packet in backward direction
bwd pkt len mean	Mean size of the packet in backward direction
bwd pkt len std	Standard deviation size of the packet in backward direction
flow bytes/s	Number of flow bytes per second
flow pkts/s	Number of flow packets per second
flow IAT mean	Mean time between two packets sent in flow
flow IAT std	Standard deviation time between two packets sent in flow
flow IAT max	Maximum time between two packets sent in flow
flow IAT min	Minimum time between two packets sent in flow
fwd IAT min	Minimum time between two packets sent in forward direction
fwd IAT max	Maximum time between two packets sent in forward direction
fwd IAT mean	Mean time between two packets sent in forward direction
fwd IAT std	Standard deviation time between two packets sent in forward direction
fwd IAT tot	Total time between two packets sent in forward direction
bwd IAT min	Minimum time between two packets sent in backward direction
bwd IAT max	Maximum time between two packets sent in backward direction
bwd IAT mean	Mean time between two packets sent in backward direction
bwd IAT std	Standard deviation time between two packets sent in backward direction
bwd IAT tot	Total time between two packets sent in backward direction
fwd PSH flag	Number of times the PSH (Push) flag was set in packets traveling in the forward direction (0 for UDP)
bwd PSH flag	Number of times the PSH (Push) flag was set in packets traveling in the backward direction (0 for UDP)
fwd URG flag	Number of times the URG (Urgent) flag was set in packets traveling in the forward direction (0 for UDP)
bwd URG flag	Number of times the URG (Urgent) flag was set in packets traveling in the backward direction (0 for UDP)
fwd hdr len	Total bytes used for headers in the forward direction
fwd pkts/s	Number of forward packets per second
bwd pkts/s	Number of backward packets per second
min pkt len	Minimum length of a packet
max pkt len	Maximum length of a packet
pkt len mean	Mean length of a packet
pkt len std	Standard deviation length of a packet
pkt len var	Variance length of a packet
FIN flag count	Number of packets with FIN (Finish) flag set
SYN flag count	Number of packets with SYN (Synchronize) flag set
RST flag count	Number of packets with RST (Reset) flag set
PSH flag count	Number of packets with PSH (Push) flag set
ACK flag count	Number of packets with ACK (Acknowledgement) flag set
URG flag count	Number of packets with URG (Urgent) flag set
CWR flag count	Number of packets with CWR (Congestion Window Reduced) flag set
ECE flag count	Number of packets with ECE (Explicit Congestion Notification Echo) flag set
down/up ratio	Download to upload ratio
avg pkt size	Average size of packet
avg fwd seg size	Average segment size observed in forward direction
avg bwd seg size	Average segment size observed in backward direction
fwd avg bytes/bulk	Average number of bytes bulk rate in the forward direction
fwd avg packets/bulk	Average number of packets bulk rate in the forward direction
fwd avg bulk rate	Average bulk rate in the forward direction

Table A.3: List of CICIDS2017 Features (contd.) [8]

Feature	**Description**
bwd avg bytes/bulk	Average number of bytes bulk rate in the backward direction
bwd avg pkts/bulk	Average number of packets bulk rate in the backward direction
bwd avg bulk rate	Average bulk rate in the backward direction
subflow fwd pkts	average number of packets in a sub flow in the forward direction
subflow fwd bytes	average number of bytes in a sub flow in the forward direction
subflow bwd pkts	average number of packets in a sub flow in the backward direction
subflow bwd bytes	average number of bytes in a sub flow in the backward direction
init win bytes fwd	total number of bytes sent in initial window in the forward direction
init win bytes bwd	total number of bytes sent in an initial window in the backward direction
act data pkt fwd	Count of packets with at least 1 byte of TCP data payload in the forward direction
min seg size fwd	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 active

Table A.4: CICIoT2023 Dataset Class Description [9]

Class	**Total Flows**	**Percentage**	**Attack Type**	**Description**
Normal	1098195	2.35%	Non malicious data	This represents data from normal user activities in the IoT network.
DDoS	33984560	72.80%	Volumetric and HTTP vulnerability exploit attack (Attack on availability)	DDoS attack is an attack that targets the availability of resources in the IoT network. This class of attack is a combination of various types of Distributed Denial of Service (DDoS) attacks such as ICMP flood, UDP/TCP flood, PSHACK flood, SYN flood, RSTFIN flood, SynonymousIP flood, HTTP flood, UDP Fragmentation, ACK Fragmentation, ICMP Fragmentation, and Slowloris. These attacks are targeted at Internet of Things (IoT) devices within the network.
DoS	8090738	17.30%	Volumetric and HTTP vulnerability exploit attack (Attack on availability)	DoS attack in an IoT network is a type of volumetric attack primarily focused on disrupting the availability of IoT operations. DoS attack class is a combination of several DoS attacks such as HTTP flood, UDP/TCP flood, and SYN Flood.
Mirai	2634124	5.64%	Volumetric attack (Attack on availability)	Mirai attack is a volumetric DDoS attack targeted on multiple IoT devices within the network. This attack class consists of multiple varieties of Mirai attacks such as the GREETH/GREIP flood, and UDP Plain flood.
Spoofing	486504	1.04%	Masquerade attack (Attack on confidentiality)	In a spoofing attack, the attacker poses as a genuine entity within the network to illicitly obtain access. This category of attack encompasses attacks such as ARP Spoofing and DNS spoofing.
Reconnaissance	354565	0.79%	Probing attack (Attack on confidentiality)	In this attack an unauthorized entity tries to actively acquire information about a target before launching an actual attack. This class of attack consists of various attack techniques such as Ping Sweep, OS Scan, Port Scan, Vulnerability Scan, and Host Discovery.
Web attack	24829	0.05%	Vulnerability exploit attack (Attack on integrity)	Web based attacks in an IoT network exploit the vulnerabilities of web applications/services associated with IoT devices. This class of attacks consists of several common types of web attacks such as SQL injection, Command injection, Uploading attack, Cross-Site Scripting, Backdoor Malware, and Browser Hijacking.
Brute Force	13064	0.03%	Manual, trial and error attack (Attack on confidentiality)	A Brute Force attack in an IoT network is a cybersecurity attack where an attacker can target the authentication mechanism of an IoT device by trying all possible combinations of the user id and passwords. This attack class includes attacks such as dictionary brute force.
Total	46686579	100%	-	-

Table A.5: List of features in CICIoT2023 dataset [9]

Feature	**Description**
Flow duration	Duration of the flow for the packet.
Header length	Length of the header
Protocol type	Type of protocol employed such as UDP, TCP, ICMP, IP, IGMP, Unknown
Duration	Time to Live (TTL)
Rate	The rate at which the packets are being sent within a network flow.
Srate	The rate at which the packets are being transmitted from source to destination (outbound) within a network flow.
Drate	The rate at which the packets are being received by a destination from a source (inbound) within a network flow.
fin_flag_number	The value of FIN (Finish) flag. Can be set to 0 or 1.
syn_flag_number	The value of SYN (Synchronize) flag. Can be set to 0 or 1.
rst_flag_number	The value of RST (Reset) flag. Can be set to 0 or 1.
psh_flag_number	The value of PSH (Push) flag. Can be set to 0 or 1.
ack_flag_number	The value of ACK (Acknowledgement) flag. Can be set to 0 or 1.
ece_flag_number	The value of ECE (Explicit Congestion Notification Echo) flag. Can be set to 0 or 1.
cwr_flag_number	The value of CWR (Congestion Window Reduced) flag. Can be set to 0 or 1.
ack_count	The number of packets within a network flow where the ACK flag is set.
syn_count	The number of packets within a network flow where the SYN flag is set.
fin_count	The number of packets within a network flow where the FIN flag is set.
urg_count	The number of packets within a network flow where the URG flag is set.
rst_count	The number of packets within a network flow where the RST flag is set.
HTTP	Indication that the application layer protocol is HTTP.
HTTPS	Indication that the application layer protocol is HTTPS.
DNS	Indication that the application layer protocol is DNS.
Telnet	Indication that the application layer protocol is Telnet.
SMTP	Indication that the application layer protocol is SMTP.
SSH	Indication that the application layer protocol is SSH.
IRC	Indication that the application layer protocol is IRC.
TCP	Indication that the transport layer protocol is TCP.
UDP	Indication that the transport layer protocol is UDP.
DHCP	Indication that the application layer protocol is DHCP.
ARP	Indication that the data link layer protocol is ARP.
ICMP	Indication that the network layer protocol is UDP.
IPv	Indication that the network layer protocol is IP.
LLC	Indication that the data link layer protocol is LLC.
Tot sum	Total combined length of all the packets within a network flow.
Min	The size of the smallest packet within a network flow.
Max	The size of the largest packet within a network flow.
AVG	The average size of the packet within a network flow.
Std	Standard Deviation of packet length within the network flow.
Tot size	Total size of the packet.
IAT	Inter Arrival Time
Number	Total count of the packets for a network flow.
Magnitude	(Average of the lengths of received packets in the flow + average of the lengths of transmitted packets in the flow)$^{0.5}$
Radius	(Variance of the lengths of received packets in the flow + variance of the lengths of transmitted packets in the flow)$^{0.5}$
Covariance	Covariance of the lengths of received and transmitted packets
Variance	(Variance of the lengths of received packets in the flow/ variance of the lengths of transmitted packets in the flow)
Weight	(Total packets received x Total packets transmitted).

Table A.6: List of acronyms

Acronym	Description	Acronym	Description
AA	Atypical Attack	GAN	Generative Adversarial Network
AAE	Adversarial Autoencoder	GB	Gradient Boosting
AC	Actor Critic	GBT	Gaussian Based Thresholding
AE	Autoencoder	GEA	Generating Evolution Algorithm
ADASYN	Adaptive Synthetic Sampling	GNB	Gaussian Naive Bayes
Adv-PA	Adversarial Polymorphic Attack	GS-HPO	Grid Search Hyperparameter Optimization
AI	Artificial Intelligence	HFSE	Heterogeneous Feature Selection Ensemble
AIDS	Anomaly-based Intrusion Detection System	HIDS	Host-based intrusion detection system
ANN	Artificial Neural Network	HOIC	High Orbit Ion Cannon
ANOVA	Analysis of Variance	HP	Hyperparameter
AUC	Area Under the Curve	HPO	Hyperparameter Optimization
BA	Balanced Accuracy	HTTP	Hypertext Transfer Protocol
BE	Backward Elimination	IDS	Intrusion Detection System
BiGAN	Bidirectional Generative Adversarial Network	IoT	Internet of Things
BiLSTM	Bidirectional Long Short-Term Memory	IPS	Intrusion Prevention System
BIM	Basic Iterative Method	JSD	Jensen–Shannon Divergence
BNB	Bernoulli Naive Bayes	JSMA	Jacobian-based Saliency Maps Attacks
bs	Batch Size	KLD	Kullback–Leibler Divergence
CNN	Convolutional Neural Network	KNN	k- Nearest Neighbor
CGAN	Conditional Generative Adversarial Network	KS Test	Kolmogorov–Smirnov Test
CVAE	Conditional Variational Autoencoder	LGBM	Light Gradient Boosting Machine
CVAE-AN	Conditional Variational Autoencoder- Adversarial Network	LIME	Local Interpretable Model-Agnostic Explanations
C& W	Carlini& Wagner's attack	LR	Logistic Regression
DAE	Deep Autoencoder	lr	Learning Rate
DAEQ-N	Deep Autoencoder Q-network	LSTM	Long Short-Term Memory
DBN	Deep Belief Network	LSVC	Linear Support Vector Classifier
DCGAN	Deep Convolutional Generative Adversarial Network	MAPE-K	Monitor-Analyze-Plan-Execute over a shared Knowledge
DDoS	Distributed Denial of Service	ML	Machine Learning
DDQN	Double Deep Q-network	MLP	Multi-Layer Perceptron
DHR-Net	Deep Hierarchical Reconstruction Network	MS-HPO	Manual Search Hyperparameter Optimization
DL	Deep Learning	NB	Naive Bayes
DNN	Deep Neural Network	NIDS	Network-based Intrusion Detection System
DoS	Denial of Service	OCN	Open-set Classification Network
DQN	Deep Q-Network	OC-SVM	One-Class Support Vector Machine
DRL	Deep Reinforcement Learning	OER	Overall Error Rate
DT	Decision Tree	PA	Polymorphic Attacks
DTC	Decision Tree Classifier	PCA	Principal Component Analysis
ES-HPO	Evolutionary Search Hyperparameter Optimization	PG	Policy Gradient
ESR	Evasion Success Rate	pID	IDS Proficiency
ETC	Extra Tree Classifier	QQ Plot	Quantile Quantile Plot
EVT	Extreme Value Theory	RFC	Random Forest Classifier
FA	Factor Analysis	RFE	Recursive Feature Elimination
FAR	False Alarm Rate	ROC	Receiver Operating Characteristic
FC-Net	Fully Connected Network	ROS	Random Oversampling
FGSM	Fast Gradient Signed Method	RUS	Random Undersampling
FN	False Negative	SAE	Stacked Autoencoder
FNR	False Negative Rate	SDN	Software Defined Network
FP	False Positive	S-DTC	Stacked Decision Tree Classifier
FPR	False Positive Rate	SGAN	Semi-supervised Generative Adversarial Network
FS	Feature Selection	SHAP	SHapley Additive exPlanations
GA	Genetic Algorithm	SIDS	Signature-Based Intrusion Detection
GACN	Generative Adversarial Cooperative Network	SMBO	Sequential Model-Based Optimization

Table A.7: List of acronyms (contd.)

Acronym	Description	Acronym	Description
SMOTE	Synthetic Minority Oversampling Technique	TPR	True Positive Rate
SSAAE	Semi-supervised Adversarial Autoencoder	t-SNE	t-Distributed Stochastic Neighbor Embedding
SVM	Support Vector Machine	UA	Unknown attacks
TA	Typical Attack	VAE	Variational Autoencoder
TL	Transfer Learning	WGAN	Wasserstein Generative Adversarial Network
TN	True negative	WGAN-GP	Wasserstein Generative Adversarial Network with Gradient Penalty
TNR	True negative Rate	WnD	Wide and Deep
TP	True Positive	XAI	Explainable AI algorithm