Application Specific NIDS

(using unsupervised techniques)

Thesis submitted to the
Indian Institute of Technology Guwahati
for the award of the degree

of

Doctor of Philosophy

in

Computer Science and Engineering

Submitted by

Ritesh Ratti

Under the guidance of

Prof. Sukumar Nandi & Prof. Sanasam Ranbir Singh

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

June, 2024

Dedicated to

My beloved Parents

and

My Family

Who always picked me up on time
and encouraged me to go on every adventure,
specially this one

Declaration

I certify that:

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisors.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.

Date: June, 2024
Place: Guwahati

(Ritesh Ratti)

Acknowledgements

This thesis is a result of a combination of efforts of so many people who have helped me directly or indirectly during my exciting journey toward a Ph.D.

I take this opportunity to thank my supervisors, *Prof. Sukumar Nandi* and *Prof. Sanasam Ranbir Singh* for their continuous encouragement, loving-kindness, and guidance during this thesis work. It has been a long association that we have carried and there are many inspiring and encouraging moments. The experience with them taught me valuable things like patience and helping fellow people. They have given me a free hand and allowed me sufficient time to think on my own. They never forced me or chained me with deadlines, albeit always made me aware of the clock ticking away. They are just a call away whenever I feel like discussing my research progress with them. Apart from technical discussions related to PhD, we had many candid discussions on various topics which I will always cherish.

It also gives me pleasure to convey my gratitude to the doctoral committee members Prof. Ashish Anand, Dr. Amit Awekar, and Dr. Vijay Saradhi for continuously advising me throughout the Ph.D. program. I would like to express my earnest gratitude to all of them for their invaluable time in evaluating the work progress and for fruitful pieces of advice towards improving the work quality. I never felt any professional pressure in our conversations. This credit goes to them for the cordial relationship that they maintain. I would like to thank all the faculty members and technical staffs of the Department of CSE for their extended help and support. Their friendly approach towards students are appreciable.

I also acknowledge my friends P. Bhagath, Pradeep Kumar Bhale, Dhrubajyoti Pathak, Swarup Ranjan and Nanu Alan Kachari for supporting me throughout

the journey in various ways. I appreciate the meaningful discussion we had during the whole tenure. Despite being far from them, they were always ready to help me out and were available in no time.

Without having adequate support from family members, traveling the doctoral journey could only remain a dream. I am grateful to all my family members for their support. the most of all for my loving, supporting, encouraging, and patient wife Megha whose faithful support during the final stages of the Ph.D. is most appreciated. And to my little angels Sayesha and Mishka, who are such nice kids that I could comfortably complete my thesis even though they are newly born. Lastly, I must thank my parents for being an inspiration to finish the thesis. The highest amount of acknowledgment goes out to them. Their sacrifices ensured that no obstacles could hinder the pace of my journey. Thank you, Mother (Mrs. Neelkamal Ratti) and Father (Mr. Virendra Kumar Ratti) for being so very kind, supportive, and caring and for all your love. This thesis is a culmination of your blessings! Last, but not least I would like to thank Almighty for his grace.

Certificate

This is to certify that this thesis entitled **Application Specific NIDS (using unsupervised techniques)** being submitted by Ritesh Ratti to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bonafide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy (Ph.D.) of the Institute. The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

Prof. Sukumar Nandi
Department of Computer Science and Engineering
IIT Guwahati

Prof. Sanasam Ranbir Singh
Department of Computer Science and Engineering
IIT Guwahati

List Of Acronyms

IDS	Intrusion Detection System
NIDS	Network based Intrusion Detection System
HIDS	Host based Intrusion Detection System
IPS	Intrusion Prevention System
ANN	Artificial Neural Network
DNN	Deep Neural Network
MLP	Mulitlayer Perceptron
RNN	Recurrent Neural Network
LSTM	Long short term memory
Bi-LSTM	Bi-directional Long short term memory
SVM	Support Vector Machine
OC-SVM	One class Support Vector Machine
ReLU	Rectified Linear Unit
DNS	Domain Name System
AE	Auto Encoder
CNN	Convolutional Neural Network
MSE	Mean Squared Error
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
FP	False Positive
FN	False Negative
TP	True Positive
TN	True Negative
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Socket Shell
FTP	File Transfer Protocol
DoS	Denial of Service
DDoS	Distributed Denial of Service
CIC	Canadian Institute for Cybersecurity
NIC	Network Interface Card
CVE	Common Vulnerability Exposure
SDN	Software Defined Networking
GAN	Generative Adversarial Networks

Abstract

In recent years, the use of unsupervised learning-based methods for network intrusion detection has attracted much attention. Multiple methods using unsupervised mechanisms have been proposed that utilize the information in various formats like network packets, flow information, etc., and use various methods for attack identification. Most of these methods have the limitations on not considering the time factor inherently but explicitly using the time-dependent features for various time windows and considering equal importance for all previous contexts. Also ignoring the fact that each protocol-specific attack is unique and ignoring the protocol awareness to determine attacks. Moreover, considering a single type of view or set of features (network header or flow) to build a machine learning model and ignoring the importance of different views in attack determination. This thesis presents four unsupervised learning-based methods in this direction.

The first method proposes a network intrusion detection method by using a Time-aware LSTM autoencoder that uses the concept of regeneration error estimation on contextual data and predicts an attack if it is higher than the defined threshold. The introduction of Time-aware LSTM units in the autoencoder network considers the time decay inherently based on previous contextual information and leads to better overall metrics in comparison to other units like MLP / LSTM. The experiments show that the Time-aware LSTM-based model outperforms the baseline models. This research work further utilized the autoencoder-based method to build a protocol-aware system for attack detection. In the second method, a Protocol aware unsupervised network intrusion detection method is proposed that provides a way to incorporate protocol channel importance while deciding attacks. The concept of protocol awareness is introduced by learning the local (protocol-specific) and global representation individually by protocol channels and incorporating the channel importance by utilizing an attention network. Through the experiments on two NIDS datasets, and multiple protocol-specific attacks, it is analyzed that the proposed method outperforms the non-protocol-aware method.

In the third method, a multiview-based network intrusion detection system is developed by using a self-supervised learning-based method. In this method, the network packet level data is utilized to extract multiple possible views with predefined time windows consecutively and construct the autoencoder-based model specific to each view. Once the encoders for each specific view are created, this work proposes the strategy to build the self-supervised

learning-based model using majority voting on the classified output from individual encoders. Through the experiments on two attacks FTP brute force, and UDP DDoS attack, it is demonstrated that the proposed method outperforms the individual view-based mechanism when multiple views are utilized in computation. The last method is the online method for network attack detection by using an on-the-fly mechanism to create clusters and utilize the statistical features of computed clusters and compare the cluster profiles. Later on, the distance between cluster profiles is estimated and an attack is identified if the distance is more than the defined threshold. The proposed model executes in an online mode as it executes the algorithm for each discrete time window data generated through flows. The experiments conducted on two widely used NIDS datasets show that attack is identified with high accuracy and the model can be deployed in real network traffic easily.

Contents

List of Figures

List of Tables

1 Introduction

With the advancement of the Internet, network attacks have become more destructive and pose a great threat to overall security. The number of attacks on computer networks has increased a lot over the years. Cybercrime is expected to cause $10.5 trillion USD in global losses by 2025 [3]. According to Mcafee [4], the potential cost of cybercrime to the global community is enormous around $1 trillion and a data breach costs the average company of about $3.8 million. These attacks are easily performed by exploiting existing vulnerabilities in various networking protocols like SSH (Secure Socket Shell), FTP/FTPS (File Transfer Protocol), HTTP/HTTPS (Hypertext Transfer Protocol), ICMP (Internet Control Message Protocol), SCP (Secure Copy Protocol) etc. using sophisticated open-source utilities like LOIC [5], Slowloris [6], Nmap [7], etc. Network-related attacks consist of many types depending upon the specific vulnerability it targets for. Some of the attacks are as follows:

1. **Fingerprinting**: These groups of attacks involve the collection of information for running operating systems, running applications, open ports, etc. for the host in a network, and understanding the existing vulnerabilities. This is the initial step while attacking the host.
2. **Sniffing or Eavesdropping**: In such attacks, the attacker listens to or monitors

network traffic and tries to capture unprotected information. The attacker further uses the captured information to perform other attacks.

3. **Spoofing**: In these kinds of attacks, the attacker spoofs the legitimate host using the address information of the legitimate host and sends the spoofed messages to the victim with the purpose of impersonating a genuine host and concealing its own identity. IP Spoofing and DNS Spoofing are some types of attacks related to this category.
4. **Brute Force attacks**: With the help of these attacks, the attacker attempts to get access to the victim's resources by trying out multiple login attempts. Brute force attacks are applicable to various applications such as FTP, SSH, or HTTP.
5. **Denial of Service**: In these kind of attacks, the attacker prevents users from accessing services on the network. These attacks can also be performed in a distributed way where an attacker utilizes the compromised hosts to target the victim machine. SYN Flooding, UDP Flood, etc. are kind of these attacks.

Intrusions are these sets of actions that attempt to compromise the confidentiality, integrity, or availability of the systems. Whereas Intrusion detection systems (IDS) are essential tools to guarantee availability, confidentially, and data integrity to detect and help in avoiding these attack activities. Intrusion detection systems are hardware or software applications that monitor these malicious activities and raise the alarms to the administrator. These malicious threats have continuously emerged over the years, so the requirement for an advanced security solution is inevitable. IDS are generally confused with firewalls. The Firewall looks outwardly for intrusions to stop them before they enter the protected network. It analyzes the packet and filters incoming and outgoing traffic based on predefined rules. On the other hand, IDS only has a monitoring role and never blocks the communication. Due to these reasons, IDS is preferred over firewalls for securing modern encrypted communication and hosts.

1.1 IDS taxonomy based on placement

IDS can be classified into two types based on its placement i.e. Host-based or Network-based. Host-based IDS (HIDS) monitors and analyzes the specific host for running applications, system calls, system logs, file integrity checking, policy monitoring, etc., and generates alarms if any suspicious activity is identified. Various HIDS-based solutions like file integrity

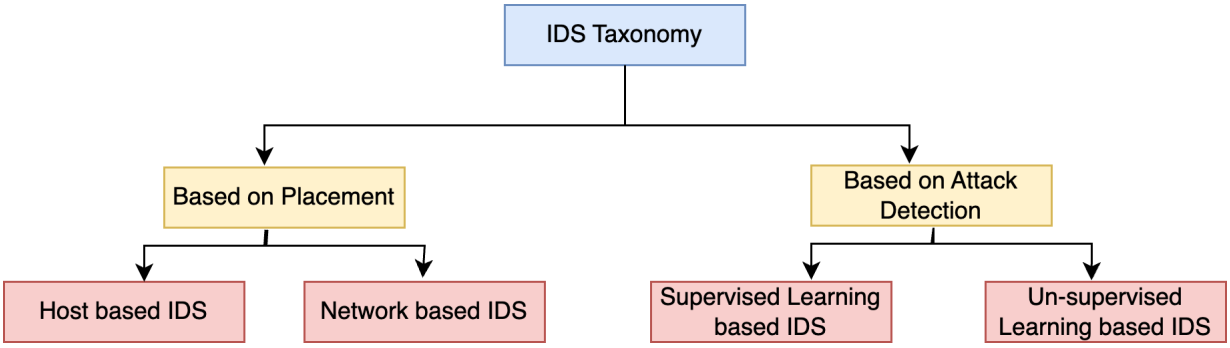

Figure 1.1: *IDS Taxonomy*

checkers are used to detect intrusions by checking the checksum of files at regular intervals. Tripwire [8], and Osiris [9] are some of the available solutions. HIDS solutions can also be implemented through log analysis where known patterns are matched in logs by using regular expressions. Open Source Security Event Correlator (OSSEC) [10] is one of the tools used for this purpose. On the other hand, Network-based IDS (NIDS) detects malicious behavior by monitoring the network traffic. Network-based IDS are normally deployed to the dedicated machine or switches that receive entire network traffic for analysis. In order to receive all network traffic the NIC card of the machine is required to run in promiscuous mode. This thesis mainly deals with developing machine learning-based solutions for NIDS.

1.2 IDS taxonomy based on attack detection

Several machine-learning approaches have been proposed in developing IDS systems for detecting various types of malicious attacks [11–13]. Such approaches may be classified into two types based on the detection method i.e. *Supervised learning based IDS* and *Unsupervised learning based IDS*.

1. **Supervised learning based methods :** Supervised learning-based methods generally use methods such as classifiers [14–16] or signature matching [17–19] or regressors [20], that consider building the models using labeled network traffic data. Signature-based approaches generally use attack-specific fixed signatures for attack detection. These signatures are stored in databases and network activity is flagged as an attack if the signature is matched. Signature matching-based utilities like Snort [21], Bro [22] etc. are used to work with these predefined sets of rules and detect attacks

based on fixed rules. Signature-based IDS can also follow fixed attack-specific signatures that are defined by multiple open-source databases like National Vulnerability Database (NVD), Common Vulnerability Exposure (CVE), Bugtraq, etc. A simple example of a Snort signature for an ICMP Flooding attack is represented as follows:

alert icmp any any − > HOME_NET any (msg: "ICMP flood";sid: 1000001; rev:1; classtype: icmp-event;detection_filter:track_by_dst,count 500, seconds 3;)

This signature is used to flag an ICMP flood attack if more than 500 ICMP packets are generated within 3 seconds. Various classifiers and regression-based methods are also proposed in this area that on the other hand require labeled data to train the model. This requires data updates frequently to consider the detection of updated and new attack scenarios.

2. **Unsupervised learning based methods :** Unsupervised learning-based methods attempt to identify anomalous patterns in unlabeled data that significantly deviate from the normal scenario [23, 24]. These kinds of IDS build a model of the normal behavior of the system and then estimate the deviations in the monitored data depending on the pre-determined threshold. Multiple methods using unsupervised learning-based mechanisms have been proposed utilizing the normal data and estimating these thresholds in terms of distance [25], entropy [26] or regeneration error estimation [27, 28]. Subsequently predicting an attack if the estimated value is greater than the learned threshold. The major advantage of these systems is that these approaches can detect unknown attacks. Whereas the major drawback is that it often generates an overwhelming number of false alarms.

Comparison between two approaches are given in table 1.1. Considering detection of unseen attacks and labeled data generation is a herculean task, this thesis proposes methods for detecting the network intrusions using unsupervised learning-based methods. The objective of this thesis is to propose solutions that improve the quality of unsupervised learning based attack detection utilizing machine learning-based techniques. In section 1.3 the current research gaps in building unsupervised learning-based IDS solutions are discussed that motivate us to build new methods for IDS solutions. Section 1.4 mentions the major objectives of this research. Section 1.5 presents the list of contributions made in this

Table 1.1: *Comparison of Supervised and Unsupervised learning based IDS*

Method	Data Requirement	Capture new attacks	Data Update	False Alarms
Supervised Learning	Labelled Data is desired that involves huge manual labelling effort.	It is unable to capture the new attacks.	Frequent data update is needed.	Less number of false alarms are generated.
Unsupervised Learning	Labelled data is not required and model can be build on normal data alone.	It can capture new attacks and zero day attacks.	It does not require frequent data updates.	High number of false alarms are generated.

direction and the organization of the thesis is presented in section 1.6

1.3 Research Challenges and Motivation

In recent years, a variety of methods [29, 30] have been devised for unsupervised learning-based IDS. However, most of these implementations have the following research gaps that motivate us to work in this direction.

1.3.1 Challenges related to data

As mentioned earlier, supervised learning-based IDS not only requires labeled network traffic data or fixed rules to start with as well as fails to identify new types of attacks and zero-day attacks. To resolve this issue various unsupervised learning-based mechanisms have been proposed. Although unsupervised learning-based methods are more effective in detecting a wider range of attacks, building these models are computationally intensive and requires clean data that is free from contamination and attack-specific information. Some other major challenges related to data are as follows:

1.3.1.1 Data Volume

The volume of network traffic and high dimensionality is another challenge for building unsupervised learning-based IDS. Most of the earlier proposed unsupervised learning-based solutions like OC-SVM [31], Isolation Forest [32], etc. are not able to create on large network

data. Recently autoencoder-based models [27, 33, 34] have been proposed to solve this issue that can work on huge network traffic. To reduce the dimensions, various methods [35] like PCA, SVD, T-SNE, etc. has also been proposed to resolve the problem of curse of dimensionality.

1.3.1.2 Data availability

On the other hand, getting labeled data for the generated network communication data is also cumbersome. Generative models like [36, 37] have been recently used to solve this issue with synthetic data. Self-supervised learning is another method in this direction that can be used to identify these intermediate labels from the structure of the data which transform the problem of unsupervised attack detection into a classification problem. Autoencoder-based models can inherently provide self-supervision by looking into the reconstruction error measures [38, 39]. This motivates us to build a self-supervised learning-based framework where the outcome of individual view-based models is utilized in generating the labels in the background and training a classification model.

1.3.2 Challenges related to type of anomaly

Unsupervised learning-based network intrusion detection systems involve identifying outstanding patterns that are referred to as anomalies or outliers. There are three types of anomalies (i) point, where an individual data point is considered anomalous, (ii) contextual, where the data point is found outlier in a specific context, and (iii) collective, where the collection of data points is considered as outliers.

Some of the solutions [40, 41] can capture pointwise anomalies by looking into individual data points and estimating statistical deviation from the normal condition. The major problem is that it may avoid capturing contextual and collective anomalies. The solutions like histogram analysis [42] and clustering can help to capture these scenarios well. Various clustering-based methods in [43, 44] have been utilized to identify the collective anomalies by comparing the individual clusters together which are usually based on similarity, and distance measures. However, these solutions are unable to capture the contextual nature of attacks well.

1.3.3 Challenges in captured information

Unsupervised learning-based IDS involves capturing the inherent information from benign network traffic. This captured information poses challenges in distinguishing between benign and attack events. Most of the recent solution captures the coarse representations that consist of capturing limited information and non-contextual information. The captured representation can have the following challenges.

1.3.3.1 Limited Information

In general, attacks specific characteristics are dependent on the inherent protocol. Most of the existing solutions ignore the importance of protocol properties while determining the attack and tend to build the global representation [14, 16, 45]. These methods build an unsupervised model for the entire cross-sectional data and do not consider the heterogeneous information related to protocols. The earlier proposed solutions consider attack detection for a group of attacks together, ignore this protocol-specific information internally, and generate representations independent of protocol during training. To include the protocol-specific decision, various supervised learning-based models [46–48] are proposed that apply filters at the protocol, host, or VM level for data reduction and further train the machine learning model on the reduced dataset that makes the solution protocol-specific but lacks protocol awareness while making a decision. These solutions tend to generate a homogeneous representation. In motivation to this, a method for protocol-aware unsupervised network intrusion detection systems is proposed that can learn the protocol-aware representations automatically and utilize this information for the final decision.

1.3.3.2 Standalone view

On the other hand, earlier studies have also considered the specific representations generated from network communication data individually and built the unsupervised learning-based model that is unable to capture the combined effect of multiple representations. These individual views consist of network, flow, and binary representations that are generated from the information available in network packets. That is network header (describing the protocol header-specific information), network payload (consisting of the data bytes transferred), or both. For detection of the attacks, each of these representations plays a different role. For example, network view represents statistical information about header fields and

is suitable for detecting high-volume attacks like DDoS. Flow level information is suitable to detect attacks for low volume attacks like password guessing. These multiple representations together can take advantage of each view and build a combined representation for accurate detection of attacks. A few recent works reported in [49, 50] try to capture the information from spatial and temporal views together for supervised learning mechanisms. This motivates us to develop unsupervised learning-based approach utilizing information from multiple views together and improving the detection results.

1.3.4 Challenges in capturing context

Various unsupervised learning based IDS have been proposed earlier to detect network attacks. These methods are normally based on data mining, statistical analysis, etc. to develop the network baseline profile and detect the attacks in case of high deviation. Earlier data mining-based methods like nearest neighbor information [51] based on the distance to the neighbor, Local Outlier factor based methods [52] care about the local outlier. Statistical machine learning based methods are also developed. Such as the chi-square method for anomaly detection, and dimensionality reduction-based method like PCA [53] have been used to project data to subspace and identify the anomaly. OC-SVM-based method [31] trains the classifier to distinguish abnormal data from normal.

1.3.4.1 Contextual behavior

The major limitation of earlier mentioned proposed methods is that they do not consider the context information into account for decision. However, utilizing context information in attack decisions plays an important role in correct predictions as attacks follow the particular life-cycle of events. To consider this, various solutions [28] have been developed recently that include the temporal context while training.

1.3.4.2 Diminishing temporal context

Although solutions have been developed recently to consider the contextual information while training but these solutions consider all the previous contexts with equal weights. So the diminishing weight factor is missing inherently in the model training process. However, in real scenarios, the most recent information imparts more importance. Another method [27] is also proposed that utilizes time-dependent features for various time windows but does

not consider the weight decay inherently in the model to provide a higher importance to recent events. This motivates us to consider the influence of diminishing temporal context inherently in model training and capturing the richer representations using unsupervised mechanisms.

1.3.5 Challenges for online scenarios

Machine learning models can be trained in two ways i.e. offline and online (on-the-fly). In offline training, models are trained on prepared datasets usually once and continuously used for prediction purposes. On the other hand, online models are trained continuously on the generated subset of data over time. The main shortcoming of the models trained in an offline manner is that it is unable to capture new attacks due to the change in data. The major problem in developing these online solutions is to detect the attack in the stipulated time. In order to solve this issue online learning-based methods have been devised that can be trained on new incoming data and predict the attack in the given time frame. Existing online solutions have been devised like Kitsune [27], ORUNADA [54] are proposed recently to train and detect the attacks on the fly. These methods suffer from high complexity and longer inference times due to the fact that Kitsune uses the ensemble of multiple autoencoder models and ORUNADA utilizes aggregated feature for computation. This motivates us to build an online solution that is unsupervised in nature and detects attacks on-the-fly with minimum time requirement.

1.4 Objectives

As discussed in the previous section 1.3, there are many challenges in building IDS solutions. However, this thesis is majorly focused on solving the challenges of labeled data requirements, handling new attacks, learning new patterns using self-supervision, etc., and is based on the following objectives.

1. Considering the earlier proposed methods incorporate context information into account by utilizing the time-dependent features with equal importance for all previous time contexts. To include the weight decay information inherently to provide higher importance to recent context, this thesis is focused on building the unsupervised learning-based model identifying network-related attacks.

2. Previous studies build the unsupervised model for entire cross-sectional network data together and do not consider heterogeneous information related to different protocols. This motivates us to consider the different protocol-specific characteristics by utilizing the global and protocol-specific local representations together for attack decisions and the major objective is to include a protocol channel for improving the attack detection metrics.
3. Most of the proposed solutions consider the single view into account like network, flow, or image and does not consider utilizing multiple views together for attack detection. Motivated by this, the unsupervised mechanism is needed that utilizes the combined information from different views in determining the attack activity and proposes the self-supervised learning-based method to transform the problem into a classification problem.
4. Existing challenges in already available complex on-the-fly training methods motivate us to develop the unsupervised learning-based method that can train on the incoming data at a particular time slot and identify the attacks in on-the-fly manner.

This thesis is mainly focused on building unsupervised mechanisms for identifying application-specific attacks in networks.

1.5 Contributions

To overcome the above-mentioned challenges this thesis proposes methods for unsupervised learning-based network intrusion detection. The contributions of this thesis are as follows.

The first contribution of the thesis work studied the influence of temporal context in building unsupervised learning-based IDS and utilized the Time-aware LSTM (TLSTM) based autoencoder to capture time-dependent contextual information and regularise the features using the TLSTM-based autoencoder. The proposed system is based on an unsupervised setup to estimate reconstruction errors and identify attacks based on a threshold obtained only from normal traffic. The recent benchmark dataset from the Canadian Institute for Cybersecurity (CISC) i.e. CICIDS2018 and CICDDoS2019 are used for evaluating the performance of the suggested method for FTP Bruteforce and UDPDoS attack and compared the results with LSTM-based autoencoder model.

The second contribution of the thesis work is the method of developing an unsupervised

protocol-aware IDS solution using the autoencoder-based model. It introduces efficient architecture to develop the IDS that can combine protocol-specific encoders with attention networks to include the importance of each protocol channel for attack identification. The attacks are identified based on a reconstruction error threshold obtained from normal traffic. The recent benchmark datasets - CICIDS2018 and CICDDoS2019 are utilized for evaluating performance of the proposed model on a wide set of attacks like BruteForce (FTP Bruteforce and SSH Bruteforce) and DDoS attacks (SYN Flood, UDP DoS, and NTP DoS). The major contribution of the proposed method is to introduce the protocol awareness during identification of attack.

A method of utilizing multiple views of network packets to build an unsupervised learning-based method using an autoencoder framework is also proposed as third contribution. The proposed method introduces self-supervision by utilizing output from view-specific encoders and developing a binary classification model for attack detection. The benchmark dataset CICIDS2018 for FTP-BruteForce and UDP DDoS attack detection is used to analyze the results. This work studied the capacities of multiview-based learning to solve the problem of attack detection on real networks and demonstrated that utilizing multiple views together for attack detection helps in improving the overall detection metrics. The main innovation of the proposed method is to utilize the different views and building self supervised learning based method.

This thesis further proposed an online method for network attack detection by utilizing statistical features generated from the flow-level data with specified time window. The proposed method is able to identify the attacks for continuous network traffic data and utilize the pre-learned attack-specific distance threshold for attack detection. The validation of the proposed method is performed on the CICIDS2018 dataset for the detection of FTP Brute Force and HTTP DDoS attacks. The major contribution of the approach is to perform the attack detection on-the-fly and utilizing the statistical features of cluster profiles.

1.6 Organization of the thesis

The thesis is organized as follows:

- Chapter 2 presents the background study on unsupervised learning-based methods for Intrusion detection systems. It also discusses the major concepts used in computer networks and related attacks, unsupervised learning based methods, state-of-the-art work

in this direction, and the available datasets for building intrusion detection systems.

- Chapter 3 provides an unsupervised method of attack detection using the Time-Aware LSTM model. The proposed method is based on autoencoder-based model development and estimating the reconstruction error threshold for attack detection. It discusses the advantage of the TLSTM unit over LSTM units in the autoencoder network. The results are compared with One class SVM-based outlier detection models.
- Chapter 4 discusses the unsupervised method for detecting attacks in a protocol-aware manner. The proposed method is based on combining the autoencoder and attention-based networks to learn and utilize the importance of individual protocol channels in attack decisions. The proposed method further improves the results from the non-protocol aware mechanism.
- Chapter 5 represents the usage of multiple views generated on network packet data for attack detection. The method is based on combining multiple views together and building a self-supervised training based model. The proposed method detects the attacks by using only raw packet data in an unsupervised manner. Network view, Flow view, and Image view are extracted and used to build the model, and the effect of different time window lengths on the attack metrics is discussed in the experimentation.
- Chapter 6 presents the on-the-fly method for network attack detection using statistical feature distance. It reviews the existing unsupervised mechanism to solve the problem and proposes a new approach to solve it using cluster distance estimation on computed statistical features. The proposed method trains the machine learning model in an online manner and detects the attacks for a given time window length.
- The concluding remarks of the thesis and future perspectives are discussed in Chapter 7.

2
Background Study

Intrusion detection systems deal with identifying malicious activities in the network. In current scenarios, networks have become potential targets of attackers due to existing vulnerabilities in networking protocols. IDS are the systems that monitor these activities and generate respective alarms. In October 1972, the United States Air Force published a paper written by James P. Anderson [55] where the author presented a threat-based model. It introduces the idea of automating the detection of *clandestine* users and the concept of intrusion detection is dedicated to this paper. Later, Doorth Denning published a model for real-time detection in 1987 by modeling intrusion detection as an expert system (IDES) [56]. IDES uses a rule-based expert system to detect known attacks and perform statistical anomaly detection on user and network data. Over the decades, multiple methods have been devised for intrusion detection systems and research has been transformed from knowledge-based systems to machine learning-based systems. Most of the recent machine learning-based systems that are based on supervised learning require labeled data and predefined rules, therefore to handle the evolving attacks at a fast pace and to detect zero-day attack various unsupervised mechanism have been devised.

The rest of the chapter is structured in the following way. Section 2.1 presents the networking terminologies used in this research work. Section 2.2 and 2.3 discusses the machine learning-based methods for implementing supervised and unsupervised learning-based IDS

solutions. Section 2.4 discusses deep learning algorithms used in the direction of building unsupervised solutions in this thesis. Available open source datasets are introduced in section 2.5, out of which recent data sets are used for implementation and experimentation. Section 2.6 discusses metrics used for evaluating the proposed unsupervised learning-based methods.

2.1 Network terminologies and related attacks

A computer network is a set of devices connected for information exchanges. It is a combination of hardware and software that transport data from one node to another. The protocol is a set of rules defined for data exchange. The communicating entities follow this set of protocols and define what, how, and when the communication happens. These set of protocols are organised in various layers as follows.

2.1.1 TCP/IP Layer architecture

TCP/IP protocol suite [57] is the set of protocols defined to build an interconnection of networks. It provides the abstraction of the communication mechanism provided by the network. It is modeled in different hierarchical layers that define the protocol stack. A layer provides services for the layer directly above and makes use of services provided by the layer directly below it. As defined in the figure 2.1, the IP layer provides the ability to transfer data without a guarantee of reliable delivery and TCP makes use of this service to ensure reliability. TCP/IP protocol suite consists of various layers as follows.

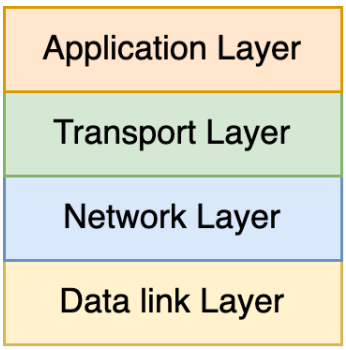

Figure 2.1: *TCP/IP layer architecture*

2.1.1.1 Data Link Layer

Data Link Layer consists of protocols that define the interface to the actual network hardware. It is responsible for the node-to-node delivery of the data and error-free transmission of data among hosts. It also performs various functions like framing to prepare the internet layer packets for transmission and uses MAC (Media Access Control) address to specify the address for transmission in the local network. MAC address consists of 48-bit representation for specifying the address. The data link layer consists of various protocols depending upon the kind of network used. For example: IEEE 802.3 Ethernet protocol is used for Local area network (LAN). MAC Flooding and MAC Spoofing are commonly known attacks in the data link layer. MAC flooding attack involves sending many packets with fake MAC addresses to overflow the switch's address table and causing it to become full and unable to process any legitimate traffic. In, a MAC spoofing attack attacker sends the spoofed messages with a changed MAC address in order to gain unauthorized access or launch a Man-in-the-Middle attack.

2.1.1.2 Network Layer

The network layer is used to send data from source network to destination network. It can also provide an unreliable transmission facility between hosts located on different networks. Internet Protocol (IP) is the well-known protocol in this layer that is information exchange mechanism used by TCP /IP protocols. It is connectionless and does not ensure reliability, flow control, or error recovery. The main purpose of this protocol is to provide routing functionality that delivers transmitted messages to the destination. The IP datagram is the basic unit of information transmitted across the networks and the IP address represents the information of the network and host in the form of a 32-bit representation in dotted format. For example: 192.168.3.4 is the IP address of the host. Various other protocols like Address resolution protocol (ARP), ICMP (Internet control message protocol), etc. also exist in this layer. ARP Cache Poisoning is a well-known attack where an attacker attempts to update the ARP table of the victim machine by sending illegitimate ARP messages. In an IP Spoofing attack, the attacker attempts to send the spoofed message with a different IP address. ICMP flooding is another well-known denial of service attack where attacker attempts to send a large volume of ICMP echo packets and overwhelm the victim machine.

2.1.1.3 Transport Layer

The transport layer provides end-to-end data transfer by delivering data from the application to the remote peer. It is responsible for the delivery of messages from one process to another process using a socket. The socket is defined as a type of file handle that is used by the process to request network services. It is a triple $< Protocol, IPAdddress, Port >$ like $< tcp, 192.168.3.4, 8080 >$. Transmission Control Protocol (TCP) is one of the protocols that is connection-oriented and provides reliable delivery, congestion control, and flow control mechanisms. Most of the application layer protocols like Telnet, FTP, etc. use it where the communication is established using a TCP connection. User datagram protocol (UDP) is another standard protocol that provides a mechanism for one application to send a datagram to another. However, it provides no reliability, flow control, or error recovery to IP. Some well-known attacks that exist in this layer are TCP SYN Flooding, UDP Flooding, etc. In SYN Flooding attack, the attacker rapidly initiates a connection to a server without finalizing the connection and exploiting the three-way handshake connection establishment procedure in TCP. In a UDP Flooding attack, attackers send large amounts of UDP traffic with spoofed IP addresses to random ports on a targeted system. Due to this, the targeted server's resources can quickly be exhausted and it becomes unavailable to normal traffic and legitimate users.

2.1.1.4 Application Layer

The application layer consists of protocols that are used by applications for communication among different hosts. Various applications have been designed to make use of these services with the help of application layer protocols. These services are utilized to make resources available and data is allowed to move and remote users can communicate well. Application layer protocols like FTP, SSH, HTTP, etc. have been defined for communication. Attacks like Brute Force attacks, DDoS attacks, etc. are exists in this layer due to existing vulnerabilities in the protocols. In brute force attacks, the attacker attempts multiple trials to crack passwords, login credentials, and encryption keys for applications like FTP, SSH, and HTTP. HTTP DDoS is another type of attack in which attackers attempt to overload a web server or application with a flood of HTTP requests causing the server to unresponsive to legitimate requests. A brief description of various application layer protocols is presented as follows.

1. File Transfer Protocol (FTP) : FTP is a standard protocol for the transfer of files among communicating hosts. It uses TCP as a transport protocol to provide reliable end-to-end delivery of files. The FTP client initiates the connection to port 21 where the FTP server is listening for new connections and follows various commands like open, pass, site, etc. to establish a connection and transfer the data. Additionally, FTP also provides security and authentication to prevent unauthorized access to the data.
2. Secure Socket Shell (SSH) : SSH is the protocol that establishes the secure connections between systems by using port 22. The traffic sent through this connection is encrypted by using public/private keys that are used to verify the both user and the remote system. It allows the application traffic with compression also.
3. Hyper Text Transfer Protocol (HTTP) : HTTP is a protocol designed to allow the traffic of HTML (Hypertext markup language) documents and is a well-known protocol for the World Wide Web (WWW). HTTP client-like browser first establishes the connection with the server at port 80 and sends the request in the form of a request method and the server responds with a status and message containing server information, body content, etc. It is a stateless protocol because it does not keep track of connections. it can also use SSL (Secure Socket Layer) as an encrypted communication channel and is defined as HTTPS and work on port 443.

2.1.2 Structure of Network Packet

A network packet is the basic unit of data that is grouped together and transferred over the network. Data sent over the computer networks is divided into packets and is recombined at the destination host. Network packets consist of three major parts the header, the payload and the trailer. The packet header consists of information about the origin and destination header that contain information from multiple protocols and the payload is the actual data that the packet is delivering to the destination. If the packet is of fixed length then the payload may be padded with blank information to make it the correct size. The trailer consists of few bits that are used in cyclic redundancy check (CRC).

Network packets go through the process of encapsulation by multiple protocols. As shown in the figure 2.2, data is divided into multiple segments where a segment is the unit data sent from the transport layer to the network layer. These segments are received at the

network layer and added to the IP header that constitutes the IP datagram. IP datagrams are further received at the data link layer and the link layer protocol-specific frame header and footer are added further that is transmitted to the destination host.

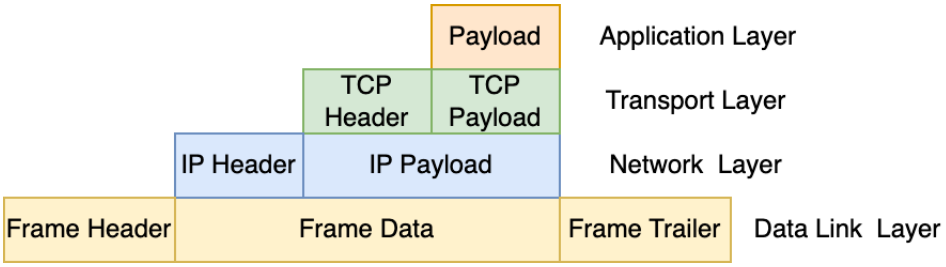

Figure 2.2: *Structure of the network packet after appending information at each layer*

2.2 Supervised learning based IDS

Supervised learning-based IDS majorly involves techniques like regression, rule-based, classification, etc. These techniques rely on network experts to provide details on characteristics and associated labels to build the signatures, rules, or classifiers.

Decision Tree-based models utilize a tree-like structure that represents classification outcomes as leaf nodes and branches are constructed based on split criteria. Various decision tree learning-based algorithms like ID3 C4.5, etc. have been proposed earlier and have been used in the development of intrusion detection systems [58, 59]. These decision trees are also used in ensemble modeling [60–62] where the final conclusion is driven by the prediction of the group of models together. Support Vector Machine (SVM) is another well-known classifier that is based on finding a separating hyperplane in feature space between two classes so that the distance between the hyperplane and the closest data points of each class is maximized. Various solutions like [63–65] are based on SVM-based learning.

Probabilistic learning-based models like the Naive Bayes algorithm is also presented where input features are assumed to be independent in which the conditional probabilities form the classifier model [66, 67]. Bayesian network ignores the independence assumption and is able to represent the variables and relationships between them. The network is constructed with nodes as the discrete or continuous random variables and directed edges as the relationships between them are used to represent a directed acyclic graph. The child nodes are dependent on their parents. Various methods like [68–70] have been proposed

that utilize the Bayesian network for classification.

Neural Network models have recently been used to develop IDS systems that are inspired by the brain and composed of interconnected artificial neurons capable of certain computations on their inputs. Various extensions of neural networks like recurrent, convolutional, feed-forward networks, etc. [71, 72] have been developed over time and have been used as a classification method for attack detection.

Regression-based methods like Logistics regression, polynomial regression, etc. are also utilized in building IDS systems [20, 73]. Since this thesis focuses on unsupervised learning-based therefore we present a detailed study of existing unsupervised learning-based methods in the next section.

2.3 Unsupervised learning based IDS

Unsupervised learning-based IDS models benign behavior of the system from the normal profile and any deviation from the known profile is considered an intrusion. The benign profile is represented as a set of features that are extracted from network packets, flows, etc. These methods generate a baseline profile of normal system activity and observations of deviations are tagged as intrusions. There are a number of machine learning-based techniques reported for unsupervised learning-based methods that can be categorized as follows.

2.3.1 Clustering based methods

These techniques attempt to group similar instances into clusters and identify the anomalous points or sets of points based on different factors like size, distance, density, etc. There are several approaches for clustering the input data. For example, connectivity models (e.g., hierarchical clustering) attempt to group the data points by the distances between them. In centroid models (e.g., k-means), each cluster is represented by its mean vector. Distribution-based models (e.g., Expectation Maximization algorithm) assume the groups to follow a statistical distribution and density-based models group the data points as dense and connected regions (e.g., Density-Based Spatial Clustering of Applications with Noise [DBSCAN]). Lastly, graph models (e.g., clique) describe each cluster as a set of connected nodes (data points).

Some of the earlier proposed methods like Syarif et al. [74] investigated the performances of various clustering algorithms when applied for anomaly detection. Five different clustering

approaches, the k-means, improved k-means, k-medoids, Expectation Maximization (EM) clustering, and distance-based anomaly detection algorithms are used for the comparison of results. Lin [75] proposed a centroid-based model using two distances for clusters that are measured and summed. The first one is based on the distance between each data sample and its cluster center, and the second distance between data and its nearest neighbor in the same cluster. This one-dimensional distance feature is used for IDS by the k-NN classifier. Here K-means clustering is used to identify cluster centers. The idea behind the proposed method is that cluster centers for a given dataset offer discrimination capabilities for the recognition of both similar and dissimilar classes. This approach provides an accuracy of 90.76%. Horng [43] proposed the connectivity model based on BIRCH clustering and built over the KDD 99 dataset for five classes (Probe, DoS, U2R, R2L). It builds five clustering feature trees where each leaf node corresponds to the cluster. After this representation dataset is obtained and feature selection is performed. 4 SVM one for each class are trained and combined together in an ensemble-based model.

Clustering-based methods are also widely used to identify brute force activity. The density-based method proposed by [44] applies the DBSCAN algorithm on FTP traffic generated at the campus network environment to separate normal data from abnormal data through clustering. It captures only selected traffic destined for FTP servers. When an attack occurs, multiple logon sessions attempt to fail, and that creates points in the same cluster with equal time series. The Indicator parameter is defined as a number of equal time intervals in the cluster. When this indicator parameter reaches a threshold value, an attack scenario is notified. Authors in [76] proposed a solution for HTTP Brute Force activities using the agglomerative approach of hierarchical clustering. The proposed approach is based on the per-connection histogram of packet payload sizes. It analyzes that the brute force attack contains three phases a) scan b) brute force c) compromise. The brute force phase features a significantly larger number of packets per flow than the scan phase. The key to identifying brute force attacks in the flow level data is to aggregate similar records into clusters and then find clustering histograms that are similar. The histograms from benign connections are quite different from histograms in typical brute-force attacks. The hierarchical clustering agglomerative approach used MDPA (Minimum difference of pair assignment) as a distance metric. In the case of non-attack traffic largest cluster may however contain histograms that are not very similar. So to filter out the estimated average intra-cluster distance for the large cluster is used to ignore if it is greater than some

threshold. Detection of brute force is done by using the largest cluster of tuples since it is assumed that attack traffic is dominant enough to compromise clusters.

The concept of a clustering-based unsupervised method for attack detection is utilized in Chapter 6 where the proposed method is implemented by using a density-based (i.e. DBSCAN) algorithm.

2.3.2 Outlier detection based methods

Outlier detection can also be used to find traffic that deviates from the baseline of normal traffic to find anomalies. Various methods like One class SVM (OC-SVM), Cluster based local outlier factor (CBLOF), Isolation Forests etc. are used to detect attacks in an unsupervised manner.

Goldstein et al. [42] proposed a very simple method using a histogram for all features and showing traffic that stands out in the histogram. The proposed approach reaches the AUC performance of 0.9999 and is able to classify the KDD99 dataset in 0.06 seconds. This might not be suitable for real data, since malicious and benign traffic is more alike. Cheng et al. [32] use a combination of isolation forest and Local Outlier Factor (LOF) to find outliers. The Isolation forest is used to find global outliers and LOF is used to find local outliers since the efficiency of LOF is lower, the performance is better. The performance of the method of Cheng et al. gets varying results. On the most imbalanced dataset, the accuracy is 0.9999 and the f1 score is 0.7692, which is due to a lower TPR, but a very low FPR. However, on a more balanced dataset, the performance is much poorer.

Payload information in network packets is also used to detect outliers. Like in the earlier proposed method PAYL [77], 256 features from the payload of every packet are used to generate the feature vector, and a model of normal system behavior is derived with the average standard deviation of these vectors. Now the outlier is detected based on the mahalanobis distance of the vector if it exceeds with predefined threshold. Another method McPAD [78] is proposed by Perdisci et al. which used One class support vector machine to classify payloads as normal or attack. It proposes a technique class 2-v grams which pairs two bytes that are v bytes apart in the payload in the packet. An iterative clustering algorithm is used to reduce the number of features and K number of One class SVM models are created. The final decision is made by the ensemble of these K models.

2.3.3 Statistical learning based methods

In statistical learning-based methods, statistical information is used as a measure for identifying deviation from normal statistical profiles in order to detect the attacks. The statistical profiles are generated by analyzing the network features and the incoming profile is marked as anomalous if deviation from the normal profile is more than the pre-defined threshold. Various statistical measures from network information like changes in the number of packets transmitted, high-frequency usage of certain IP addresses and ports, frequent changes in TCP flags, etc. can be used to learn the deviation. The statistical methods can also utilize entropy measures to analyze various network feature values for attack detection. The major drawback of statistical methods is that they fail to detect attacks if the attacker generates disruptions below the learned threshold.

Ye and Chen [40] proposed an anomaly detection technique based on a chi-square statistic in which a profile of normal events in an information system is generated. The basic idea in this approach is to detect both a large departure of events from normal as anomalous and intrusions. Krügel [41] proposed a statistical processing unit for detecting anomalous network traffic, to detect rare attacks such as R2L and U2R. A metric is developed that allows the system to automatically search identical characteristics of different service requests. The anomaly score of a request is calculated based on the three main characteristics i.e. the type of request, the length of the request, and the payload distributions. The network administrator defines a threshold to raise alarms for anomalous requests.

Manikopoulos [79] proposed a hierarchical multitier multiwindow statistical anomaly detection technology to monitor network traffic parameters simultaneously, using a real-time probability distribution function(PDF) for each parameter. The similarity measurements of measured PDF and reference PDF are combined into an anomaly status vector classified by a neural network. This methodology detects attacks and soft faults with traffic anomaly intensity as low as 3 to 5 percent of typical background traffic intensity, thereby generating an early warning. Another method presented by Tan et al. [80], for DoS attack detection is a system that uses multivariate correlation analysis (MCA) for accurate network traffic characterization by extracting the geometrical correlations between network traffic features. The proposed system involves extraction of the geometrical correlations between network traffic features. First basic features are generated in a well-defined time interval with a triangle area map generation module and then correlations between two distinct features

within each traffic instance are extracted. At last decision for attack is made based on the computed statistics.

2.3.4 Deep learning based methods

Deep Learning is the sub-field of machine learning that uses a cascade of many layers for processing where each successive layer uses output from the previous layer. In deep learning, higher-level features are derived from lower-level features after processing through multiple layers. Various unsupervised deep learning-based models like auto-encoders, etc. have been used to develop the IDS methods. The detailed description of various deep learning algorithms used in this thesis is described in section 2.4.

2.4 Deep Learning Algorithms

This section describes the deep learning algorithms in detail that are used to build the unsupervised learning-based methods in this research. It also discusses the associated research papers that are based on specific deep-learning algorithms.

2.4.1 Multi-Layer Perceptron (MLP)

A Multilayer perceptron (MLP) is a fully connected layer of perceptron units. The structure of a multi-layer perceptron consists of multiple layers as shown in figure 2.3. It consists of an input layer, one or several hidden layers, and an output layer that consists of multiple perceptron units where each perceptron unit acts as a neuron and uses a nonlinear activation function. The neurons are computation entities, that calculate activation function at each layer in a feed-forward fashion, and error is propagated backward for weight normalization. The gradient descent algorithm [81] is used to adjust and learn the weights where increment or decrement in the weight vector happens by the input vector scaled by the residual error and the learning rate. Other optimization algorithms have also been proposed recently like Adagrad (2011), Adam (2015), etc. that are more flexible with adaptive learning rates, and show impressive results when applied to deep neural networks. Multilayer Perceptron considers each input separately while processing, due to which it is unable to capture the temporal context.

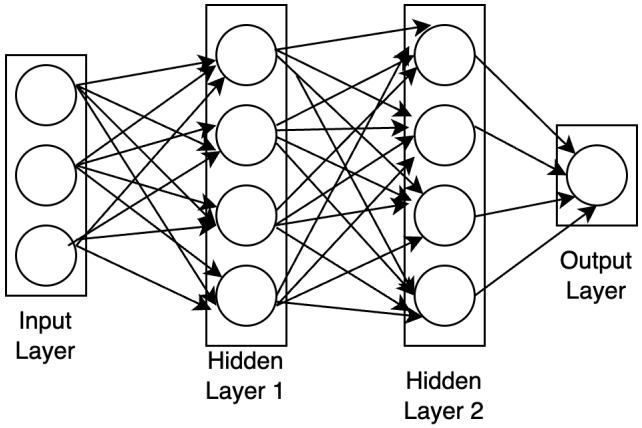

Figure 2.3: *Multi-layer perceptron*

Multilayer perceptron has been majorly used for classification purposes where the last layer contains the number of neurons depending on the number of attack classes to identify. Huang [82] proposed a new artificial neural network (ANN) called an extreme learning machine (ELM). The ELM is a single hidden layer feed-forward neural network, which randomly uses the input weights and hidden layer bias without tuning and determines the output weights in an analytical way. Li [83] proposed a Fast Learning Network (FLN) that is based on connecting the multilayer feed-forward neural network and a single-layer feed-forward neural network in parallel. FLN showed reasonable performance and stability using a smaller number of hidden nodes and utilizing less time. MLP has also been used in building an autoencoder-based network which is based on an unsupervised learning mechanism that attempts to learn the representation from benign data and identify attacks based on deviation. A recent study [84] attempts to build an autoencoder network based on MLP units to identify the anomalies. This concept has also been used in chapter 3 to compare the influence of temporal effect with the proposed autoencoder-based model. MLP-based autoencoders have also been developed in the proposed multiview-based system in chapter 5.

2.4.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNN) is the deep learning algorithm where connection patterns are inspired by the visual cortex, and is based on convolution operation. The convolutional layers parameters are a set of learnable filters where every filter is applied

along width and height of input (2-dimensional) vector. During the forward pass, each filter slides across the width and height of the input, producing a 2-dimensional activation map. As filter slide across the input, the dot product is computed between the filter and the input in a feed-forward fashion. As shown in figure 2.4, the CNN model consists of multiple operations like padding where extra information is added to consider the information from the borders that can lead to good accuracy. Pooling operation is used to reduce the number of learnable parameters and different pooling operations like max-pooling, average-pooling, etc. are applied for the set of values in the filter.

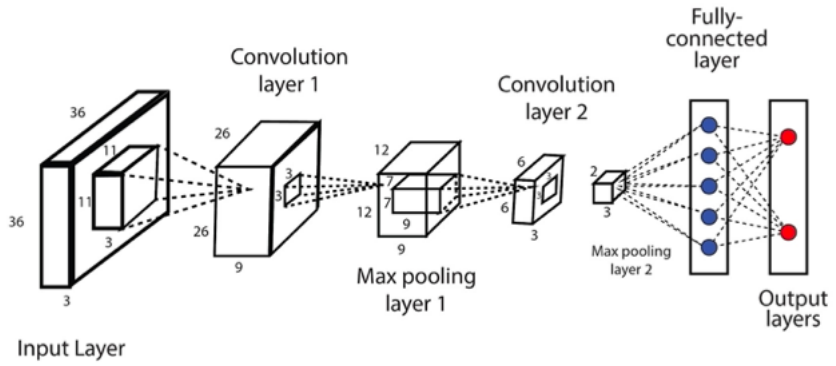

Figure 2.4: *Convolutional Neural Network*

Kim et al. [85] proposed a CNN model for detection against denial of service attacks and evaluated its performance through comparison with a Recurrent Neural Network (RNN) model. Furthermore, the optimal CNN design for better performance through numerous experiments is also suggested. Zhang [86] proposed a complex multilayer IDS model based on CNN and gcForest. It also discusses a novel P-Zigzag algorithm for converting the raw data into two-dimensional greyscale images. They used an improved CNN model in a coarse grain layer for initial detection. Then in the fine-grained layer, gcForest is used to further classify the abnormal classes into N-1 subclasses. They used a dataset by combining UNSW-NB15 and CICIDS2017 datasets. The experimental results show that the proposed model significantly improves the accuracy and detection rate compared to the single algorithms while reducing the false alarm rate. Jiang [87] proposed an efficient IDS system by combining CNN and bidirectional long short-term memory (BiLSTM) in a deep hierarchy. The class imbalance problem is addressed by using the SMOTE [88] to increase the minority samples,

which helps the model to fully learn features. CNN is used for extracting spatial features while BiLTSM is used for temporal features. The experiments are performed using NSL-KDD and UNSWNB15 datasets. CNN model has also been used in building unsupervised models like autoencoders. The proposed method [89] involves building variational autoencoder using CNN units. The similar concept has been used to build the CNN autoencoder for image views in chapter 5.

2.4.3 Recurrent Neural Network (RNN)

A Recurrent Neural Network is a neural network with directed cycles where nodes send feedback signals to each other. It is based on a recursive operation where the output of the next layer becomes input to the previous layer and RNNs are able to capture patterns for temporal context. It is constrained by shared weights across neurons where each neuron observes different times. Recurrent Neural Networks are the natural way to model sequential data. Back-propagation with time is the common strategy to learn in recurrent neural networks as shown in the figure 2.5 and the recurrent process is representation by equation (2.1) where, y_t is output at time t, y_{t-1} is the output at previous time $t-1$, x_t is the input at time t and U, W are weight matrices of the neuron.

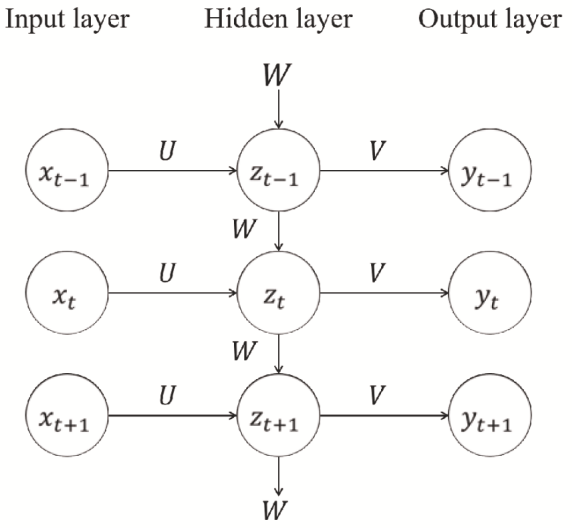

Figure 2.5: *Recurrent Neural Network with one hidden layer*

$$y_t = f(U.x_t. + W.y_{t-1}) \tag{2.1}$$

Recurrent neural networks suffer from exploding and vanishing gradient issues where the gradients shrink exponentially due to smaller weights and grow exponentially due to larger weights because of their recursive nature of processing. RNN-based IDS proposed by Yin et al [90] in the context of binary and multi-class classification of the NSL-KDD dataset. The model tested using a different number of hidden nodes and learning rates and the results showed that different learning rates and the number of hidden nodes affect accuracy of the model. The best accuracy is obtained by using 80 hidden nodes and a learning rate of 0.1 and 0.5 for binary and multi-class scenarios respectively. Xu [91] proposed an IDS based on RNN using GRU as the main memory together with the multilayer perceptron and a softmax classifier. The proposed methodology tested using KDD Cup99 and NSL-KDD datasets. The experimental results showed good detection rates in compariosn to other methodologies. The major drawback of their model is lower detection rates for minority attack classes like U2R and R2L.

2.4.4 Long Short term memory (LSTM)

Long short-term memory (LSTM) [92] is a type of recurrent neural network (RNN) that inherently captures the order sequence. It is introduced by Hochreiter and Schmidhuber in 1997 that mitigates the vanishing and exploding gradient problem by replacing the conventional nodes in the hidden layer of conventional RNN(Recurrent Neural Network) with memory cells. LSTM consists of various cells i.e. update $u_g^{<t>}$, output $o_g^{<t>}$, forget gates $f_g^{<t>}$. where these cells remember values over arbitrary time intervals and regulate the flow of information through the cell as shown in figure 2.6. Due to this reason it is used to handle vanishing and exploding gradient problems in RNN.

As shown in figure 2.6, Let us assume for given sequence of inputs x_1, x_2, . . x_t, model computes sequence of output as y_1 ,y_2. . . .y_t. Let w_f, w_u, w_o, w_a, w_d and b_f, b_u, b_o, b_a, b_d are trainable parameters (weights and bias), $c^{<t-1>}$ and $c^{<t>}$represents previous and next cell states respectively. Equations (2.2) are used to compute the forget $f_g^{<t>}$, update $u_g^{<t>}$ and output gate $o_g^{<t>}$ states and ultimately the final cell state $c^{<t>}$ and activation values $a^{<t>}$ are computed.

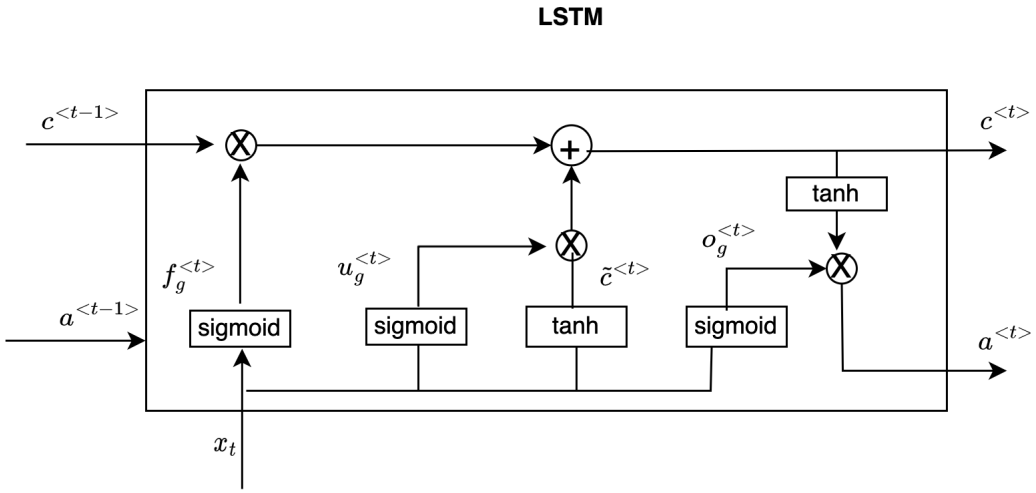

Figure 2.6: *Long short-term memory network*

$$\begin{cases} f_g^{<t>} = sigmoid(w_f[a^{<t-1>}, x^t] + b_f) \\ u_g^{<t>} = sigmoid(w_u[a^{<t-1>}, x^t] + b_u) \\ o_g^{<t>} = sigmoid(w_o[a^{<t-1>}, x^t] + b_o) \\ \tilde{c}^{<t>} = tanh(w_c x^t + u_c a^{<t-1>} + +b_c]) \\ c^{<t>} = f_g^{<t>} c^{<t-1>} + u_g^{<t>} \tilde{c}^{<t>} \\ a^{<t>} = tanh(w_a[a^{<t-1>}, x^t] + b_a) \\ a^{<t>} = o_g^{<t>}.tanh(c^{<t>}) \end{cases} \tag{2.2}$$

The fundamental idea behind the use of LSTM for anomaly detection is the system checks the past values over a certain amount of time and tries to predict the behavior for the upcoming time step. If behavior in the next time step belongs to normal behaviors, then it is normal, and anomaly otherwise as used in [93]. The Stacked LSTM RNN is also used for anomaly detection, as discussed in [94]. In this approach, the model is designed to accept only one-time steps as input, and the LSTM state is maintained across the input sequence. The data is trained with normal time series instead of training on normal data only. Thus, for each observation, there are multiple predictions made at different times previously. The prediction information is then gathered to calculate error vectors using a multivariate gaussian distribution to detect an anomaly. LSTM units have also been used to develop autoencoders that can capture the temporal effect [27, 28]. In this research, LSTM units have been used to analyze the effect of temporal features in Chapter 3. LSTM units

also also used to develop the protocol aware mechanism for IDS in Chapter 4.

2.4.5 Autoencoder

An auto-encoder(AE) is a learning algorithm that applies back-propagation, sets the target values to be equal to its inputs, and learns the compressed representation of the input data as shown in figure 2.7. The middle layer is the compressed representation. The auto-encoder model is composed of an encoder and a decoder stage. The encoder compresses the input data, causing loss of information and the decoder tries to reconstruct the compressed data as accurately as possible. The loss is then calculated as the difference between the original input and the reconstructed one. The aim of this task is to learn the intermediate representation of the input data in latent space. To compress data with minimal loss of relevant information, the network has to find patterns in the input and ideally learn some semantics or context of the data. The backpropagation and gradient descent techniques are the standard methods to train an autoencoder. An autoencoder does not require labeled input data because it uses the input as a model for correcting its output. This training process is thus unsupervised in nature. Autoencoders are particularly used for nonlinear dimensionality reductions and learned representations can be used as a classifier for attack detection. For a given input sample x, autoencoder regenerates the input $\acute{x}$ again, and can be defined in equation (2.3) where $\acute{x}$ is the regenerated output of x.

$$\acute{x} = decoder(encoder(x)) \tag{2.3}$$

For anomaly detection task, the autoecoder is generally trained using normal data, and the regeneration error (say, $e = (x - \acute{x})^2$) is expected to be low for normal data, and high for anomalous data.

Different variants of AE have been proposed like Stacked AE, Sparse AE, and Variational AE which have been used for attack detection. Autoencoder model have been used to learn the intermediate representations and use in building the classification methods. Shone et al [34] proposed an IDS based on deep AE and Random Forest. In the proposed approach, only the encoder part of AE is utilized to train the Random Forest model in order to make the model efficient in terms of computational and time. The experiments were performed for multiclass classification scenarios using KDD Cup 99 and NSL-KDD datasets. Yan [33] proposed an IDS using a stacked sparse autoencoder (SSAE) and SVM. The SSAE used as

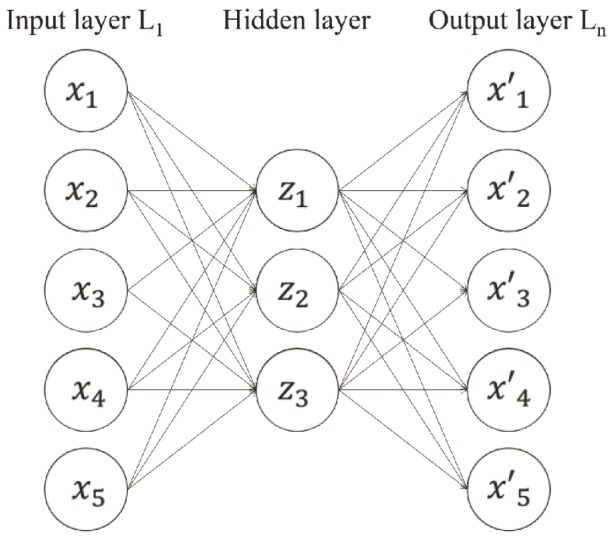

Figure 2.7: *Structure of an Autoencoder network*

the feature extraction method and SVM as a classifier. The performance results have been presented by comparing different feature selection, ML, and DL methods using the NSL-KDD dataset. Autoencoder models have also been used to build unsupervised learning-based methods by using various inherent units like MLP, LSTM, CNN, etc. These methods are usually trained on Benign traffic and identify the attacks based on high deviation from the learned thresholds. The major research contribution of this thesis is focused on building such autoencoder-based models. These methods have been extensively used in Chapter 3, Chapter 4, Chapter 5.

2.4.6 Self Attention and Multi-head Attention

LSTM units have been used earlier to build the encoder-decoder network to solve the sequence-to-sequence problem but the major drawback of the LSTM-based network is that it is unable to remember the long-range dependency because the performance of the network degrades rapidly as the length of input sequence is increased. To resolve this issue attention mechanism [95] has been devised that selectively concentrates on relevant information in the input sequence. The Self-Attention mechanism is generally used to learn the effect of different positions of sequence in order to compute the final representation. It is used in developing transformer-based models that help in the process of computing the attention

weights and is mainly used to learn the relative importance of input sequences that may result in a better intermediate representation. The importance of the attention mechanism is to highlight important concepts rather than focusing on all the information.

The self-attention is described as mapping a query (Q) and a set of key-value (K, V) pairs to an output, where the query, keys, values, and output are all vectors. The meaning of query, value, and key depends on the application for e.g. in the case of text similarity, query is the sequence embeddings of the first piece of text and value is the sequence embeddings of the second piece of text. Key (K) is usually assigned the same as a value vector. The attention is defined as in equation 2.4

$$Attention(Q, K, V) = softmax((Q.K^T).V) \tag{2.4}$$

The self-attention is usually able to capture the representations in one linear projection. However, it is beneficial to capture the representations in multiple projections, and Multi-head attention [95] is used for this. It is mainly used to generate multiple representations from the output of the self-attention layer and learns different representations using multiple heads in parallel. It combines the knowledge of the same attention pooling via different representation sub-spaces and captures richer representations. This layer is introduced to learn more generalized outcomes by learning different aspects of an input sequence. Let $head_i$ denotes the computed information in ith projection then multihead M_{head} is represented as in equation 2.5

$$\begin{cases} head_i = Attention(Q, K, V) \\ M_{head}(Q, K, V) = Concat(head_1, head_2, ...head_h) \end{cases} \tag{2.5}$$

Several solutions have been proposed by using the attention mechanism. Tan et al [96] proposed a neural attention model for intrusion detection using transformer-based architecture and tested the results on the CICIDS2017 dataset. The comparison with the bi-directional LSTM and Conditional Random Field (CRF) model suggests a meaningful improvement in the score. The concepts of self attention and multihead attention have been used in this research to learn the protocol channel importance and improving the detection metrics for unsupervised intrusion detection system. This concept is primarily used in chapter 4.

2.5 IDS Datasets

Machine learning algorithm requires a lot of data to train with and the quality of data is crucial in any machine learning problem. Over the period, several datasets have been prepared using the test bed with attacks. These datasets constitute network packets as raw data and the flow level information is extracted using various utilities and subsequently data labelling is performed by domain experts. In the intrusion detection field, a lot of standard datasets are proposed like KDD99, NSL-KDD, UNSW-NB15, etc. However, some of these datasets are too old and are unable to depict the modern network scenario. Other recent datasets like CICIDS2018 and CICDDoS2019 are specifically built to depict the modern network which is developed at CIC (Canadian Institute of Cybersecurity). These recent datasets (i.e. CICIDS2018 and CICDDoS2019) have been utilized in this thesis for building unsupervised learning-based methods that consist of realistic attack scenarios. A brief description of available datasets is as follows.

2.5.1 KDD99

KDD99 Dataset [97] is a well-known benchmark dataset in the research of Intrusion Detection techniques. It has been one of the most popular datasets since its release in 1999 which was developed by MIT Lincoln Labs. The dataset consists of nine weeks of raw TCP dump data from a simulated U.S. Air Force network, with the addition of numerous attacks. It consists of the attack categories into 4 major types i.e. Denial of Service (DoS), Probing, User to Root (U2R), and Remote to Local (R2L). Denial of Service is a kind of attack in which the attacker compromises the victims so that it becomes unable to handle legitimate requests. Syn Flooding, Smurf Attack, UDP Storm, etc., are some of the examples of DoS attacks. On the other hand, the main objective of Probing is to gather information about the victim host like port scanning, OS fingerprinting, etc. The U2R attack poses unauthorized access to local machines using superuser privileges. In this attacker tries to gain root privileges by exploiting some vulnerability in the victim machine e.g. Buffer overflow attacks. In the R2L type of attack, the attacker gains unauthorized access to the remote machine and acquires local access to the victim machine and password guessing is one kind of such attack. The KDD99 dataset contains a huge number of redundant and duplicated records and other mistakes that affect the performance of classifiers due to which it become biased towards more frequent records.

2.5.2 NSL-KDD

The NSL-KDD dataset [98] is the refined version of the KDD99 dataset which is generated after removing the existing anomalies in the KDD99 dataset. The redundant records are removed from the KDD99 dataset and it makes sure that sufficient numbers of records are available in the dataset for each category. In the complete dataset train and test files are provided separately as KDDTrain+ and KDDTest+. NSL-KDD dataset contains 41 different flow level features which are divided into 4 different types as follows:

- Basic features : duration, protocol type , service, flag, src bytes, dst bytes, wrong fragment, urgent.
- Content related features : num failed login, logged in, num compromised, root shell, su attempted, num root, num file creations, num shell, num access file, is hot login, is guest login.
- Time related features : count, srv count, serror rate, srv error rate, rerror rate etc.
- Traffic features : dst host count, dst host srv count, dst host same srv rate, dst host same src port rate etc.

Although NSL-KDD is the improved version of the KDD99 dataset, it is not a comprehensive representation of the modern attack environment. Due to this recent benchmark datasets are generated to mimic the modern network behaviour.

2.5.3 UNSW-NB15

The UNSW-NB15 dataset [99] is the latest dataset developed to overcome the limitations of the KDD99 and NSL-KDD Datasets. The UNSW-NB15 dataset consists of captures spanning two days in which attacks are launched with normal traffic for reference. This dataset defines the attack categories into 9 major types. The training dataset constitutes data in multiple CSV files. It consists of 4 different CSV files with around 1 million records. These records are collected over a test bed using Argus and Bro IDS through IXIA PerfectStorm Tool in the Cyber Range lab of UNSW Canberra. A brief introduction of different attacks is as follows.

- Reconnaissance: This is a kind of attack in which attackers gather information about the targeted machine. It can be of multiple types like IP/ Network reconnaissance, site reconnaissance, DNS reconnaissance, social engineering, port scan, OSscan, etc.

- ShellCode: It is a small piece of code that is used as a payload in the exploitation of software vulnerability. Buffer overflow attack is one of the known attacks in this category.
- Generic: Generic attacks are attacks that work against cryptographic primitives and find vulnerabilities in the available algorithms like birthday attacks, man-in-the-middle attacks, etc.
- Worm: Worm is a standalone malware computer program that replicates itself in order to spread. It uses a computer network to spread to other systems and compromise the network systems.
- Fuzzers: Fuzzers use the widest range of unexpected inputs in order to discover new and unknown vulnerabilities in the network.
- DoS: DoS attacks are majorly categorized into attacks that compromise the accessibility of the system like the Smurf attack.
- Exploits: Exploits use the widest range of unexpected inputs in order to discover new and unknown vulnerabilities in the network.
- Backdoor: Backdoor is malware that installs itself as part of an exploit and gets access to resources at the initial stage.
- Analysis: Attack involves analyzing the system information, details, stats, and other attacks that follow after this attack.

Although UNSW-NB15 is the recent dataset, it is not a comprehensive representation of the modern attacks and constitutes only a subset of the attacks. Due to this, recent benchmark datasets are generated specifically for the group of attacks that exhibit recent attacks.

2.5.4 CICIDS2018

CICIDS2018 dataset [1] is the recent dataset developed at the Canadian Institute of Cybersecurity (CIC). It presents real-time network behavior and comprises several intrusion states. It provides 10 days of traffic, from Wednesday, February 14, 2018, to Friday, March 2, 2018, and is captured on the test bed that consists of 5 different departments of 100 machines and an attack network of 50 machines as shown in figure 2.8. The dataset is developed using the

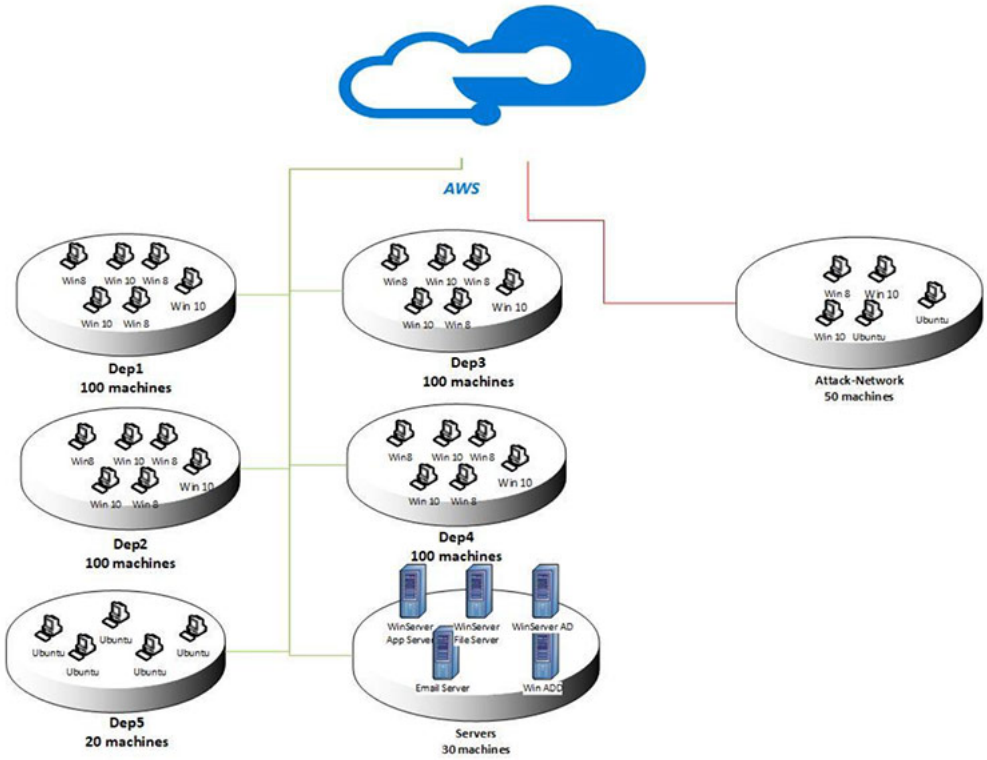

Figure 2.8: *Network Topology of CICIDS2018 dataset [1]*

concept of profiles where two general classes of profiles i.e., B-profiles are used to generate the normal traffic, and M-profiles are used for attack scenarios. FTP-Patator and SSH-Patator tools [100] are primarily used for performing brute force attacks and DDoS attacks are performed using LOIC and HOIC tools. Network packets are captured and associated flows are generated using CICFlowMeter-V3 and extracted more than 80 features. In total 7 different types of attacks are captured in the dataset on different dates. The seven different attacks include brute force (FTP-Patator and SSH-Patator), Denial of Service (slow loris, SlowHTTPTest, Hulk, GoldenEye), Heartbleed, web attacks (Damn Vulnerable Web App, XSS, brute-force), infiltration of the network from inside, botnet, and Distributed Denial of Service with port scanning (Low Orbit Ion Canon).

2.5.5 CICDDoS2019

The CICDDoS2019 dataset is published by the Canadian Institute for Cybersecurity (CIC) [2] and is publicly available which is specifically for DDoS attacks. It contains different types

of DDoS attacks that are categorized into Reflection-based and Exploitation-based DDoS attacks. In reflection-based attacks, the attacker sends the request to the victim as well as other machines in the network where the request packet carries the spoofed address of the victim, and hence all other machines reply back to the victim address causing bandwidth exhaustion. Exploitation-based DDoS attacks are possible due to a variety of vulnerabilities present in TCP/IP protocols and a set of attackers exploit these vulnerabilities to launch DDoS attacks.

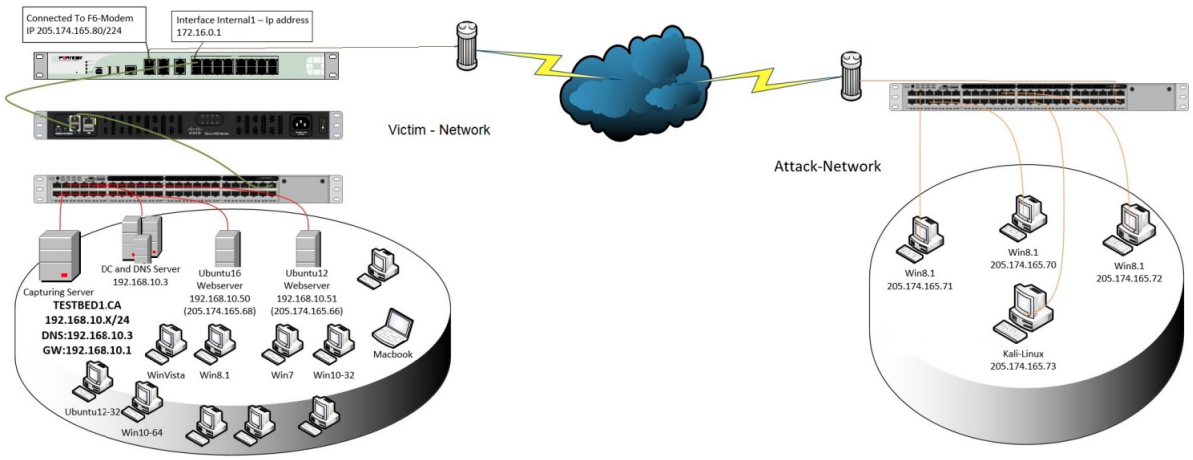

Figure 2.9: *Network Topology of CICDDoS2019 dataset [2]*

The test-bed architecture as shown in figure 2.9 consists of two completely separate networks the Attack-network and the Victim-network. The Victim-Network is secured with a firewall, router, switches, and several common operating systems however Attack-Network is a completely separated infrastructure where different attacks are executed. The victim network also consists of an agent that produces benign behaviors on each computer. The test-bed produces realistic background traffic and follows the B-Profile approach [101] which is responsible for generating benign traffic by using abstract behavior of human interactions. It employs 11 different DoS attack profiles and most of these attacks are at the application level. The dataset includes raw packets of network traffic in PCAP format and flow-level records with more than 80 features are extracted using CICFlowMeter-V3 [102].

2.6 Evaluation Metrics

To measure the effectiveness of different IDSs and machine learning models in general, a commonly used set of performance metrics has been devised to promote comparison between solutions. These metrics are used to estimate the performance of the proposed method.

2.6.1 Classification Metrics

Several classification metrics can be used to describe the performance of an IDS. Let us assume, FP denotes the False Positives, FN denotes the False Negatives, TP denotes the True Positive and TN denotes the True Negatives then various classification metrics are represented as follows :

- Precision : This is the percentage of packets detected as attacks out of total outcomes. For example, for a packet, if 7 out of 10 outcomes are identified as attacks then precision is 70 percent. This is depicted in the following equation.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

- Recall or Detection Rate : This is the percentage of packets detected as attacks out of total attacks. It is the measure of the sensitivity of the system. For example, for a packet, if 7 out of 10 are identified as True attacks then the detection rate is 70 percent. This is depicted in the following equation.

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

- F1 Score : This is the harmonic mean of precision and recall computed. A high F1 score identifies low false positives and low false negatives.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.8}$$

- Accuracy : This metric determined how much accurate the prediction is for the build model on different classes.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.9}$$

2.6.2 Clustering Metrics

To evaluate the clustering method various evaluation metrics have been proposed to estimate the effectiveness of clusters formed. A brief introduction to these metrics is as follows:

- Silhouette Score : Silhouette score is used to define how well the clusters are formed. It consists of two parts i.e. Mean distance between the sample and all other points in the same cluster and, The mean distance between the sample and all other points in the next nearest cluster. If these distances are represented as a and b then silhouette score $S_{score}$is computed as equation (2.10),

$$S_{score} = \frac{(b-a)}{max(a,b)} \tag{2.10}$$

- Purity Score : Purity score is used to identify the cluster quality where the label to each cluster is assigned based on the most frequent class and the purity is calculated based on the number of correctly matched class and cluster labels divided by the number of total data points. Let N is the number of data points, K is the number of clusters, c_i be a cluster in C, and t_j is the classification which has the max count for cluster c_i then equation (2.11) represents the purity for computed clusters,

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max_j|c_i \cap t_j|) \tag{2.11}$$

- Homogeneity Score : The homogeneity score considers that each cluster contains only members of a single class. This measure can be determined by using conditional entropy which is uncertainty in determining the right cluster given the knowledge of the class. The homogeneity score is bounded between [0,1]. A higher score denotes better clustering. Let Y_{true} and Y_{pred} are actual and predicted outcomes, and H represents the estimated entropy values. The equation (2.12) represents the formula to compute the homogeneity score.

$$Homogeneity = 1 - \frac{H(Y_{true}|Y_{pred})}{H(Y_{true})} \tag{2.12}$$

2.7 Summary

In this chapter, the details on network terminologies like the hierarchical structure of the TCP/IP layer and associated protocols and the most common related attacks are discussed. The structure of the network packet is described in terms of attached information at various stages of transfer and protocol-specific information is attached. The discussion follows with a brief explanation of supervised learning-based methods devised for intrusion detection and a detailed explanation of various unsupervised learning-based methods like clustering, outlier, statistical, and deep learning approaches is presented. Various deep learning-based processing units like MLP, CNN, LSTM, autoencoder, and self-attention networks are explained in detail and related concepts used in developing IDS are discussed. The introduction to various datasets used in IDS research is described further and provides a detailed description of features, attacks covered, etc. The chapter concludes with an explanation of various classification and clustering metrics used for evaluation. In the next chapters, the major contribution in the area of network-specific attack detection using unsupervised techniques is presented.

3

Influence of Temporal Context in Network Intrusion Detection System

3.1 Introduction

The previous chapter discusses about major concepts in computer networks, related attacks and machine learning algorithms used to build supervised and unsupervised learning-based Intrusion detection systems. The available datasets used to build the IDS systems are also discussed. Recent trends have shown that the research is focusing on building unsupervised learning-based methods for Intrusion detection. The deep learning-based methods have been utilized to learn the representations from normal data and identify the attack based on regeneration error. However, most of the proposed methods ignore the effect of decaying information with time while generating the representation.

In this chapter, the unsupervised learning based approach is proposed to detect the attack in network flows by training the model using only normal traffic and using reconstruction error as the parameter to classify the attack event. The proposed system is contextual and is able to consider the influence of temporal context while taking the attack decision. Various experiments are performed on different recent datasets like CICDDoS2019 [2], and CICIDS2018 [1] and experimental results exhibit that the proposed model overall provides

better classification metrics.

3.1.1 Significance of Temporal Context

Temporal context plays a vital role in correctly predicting the attacks due to the sequential nature of the attack life-cycle. In many of the earlier proposed methods, Long Short term memory (LSTM) [27, 28] based models have been proposed for attack detection, however such model does not consider time decay factor explicitly to provide more importance to recent events. This weight decay factor plays an important role in providing diminishing weight values for events according to time. LSTM units have also been used to build autoencoder-based networks which can be used to represent the contextual information in compressed form and able to re-generate the output same as supplied input. This capability of regeneration has been used for anomaly detection where the autoencoder is trained first with normal data and attack identification is confirmed if the regeneration error is higher than the defined threshold. Some of the earlier proposed solutions using similar concept like [27] considers unsupervised modeling using an ensemble of autoencoders and utilize the damped incremental statistics but lacks in considering the temporal context inherently as various time-specific features are pre-built with different time window length, On the other hand in solution [28] temporal context is considered using LSTM autoencoder but an inherent property of LSTM does not allow the effect of weight decay with time. This motivates us to consider elapsed time (time between consecutive events) information when predicting current events, and regularize the attribute values to a latent representation using autoencoder network. This elapsed time information is ultimately used to assign a weight decay factor to previous events. The main purpose of this research is to address this by considering the previous context with weighted decay factor using a multi-layer Time aware LSTM [103] (TLSTM), and regularize the feature and regenerate the input data using an autoencoder. In TLSTM [103], inherent units are designed to handle data with variable elapsed time between consecutive elements of the sequence and it uses subspace decomposition of all memory that enables time decay to discount memory content according to elapsed time as specified in section 3.3.2.1. TLSTM is proposed earlier to incorporate the elapsed time information into standard LSTM that enables it to capture temporal dynamics of sequential data with time irregularities.

TLSTM model has also been used earlier in the medical domain for disease progression

modeling [104] and patient subtyping [103] where information is generated in the irregular time interval. It has also been used in business process monitoring tasks as specified in [105]. With these existing implementations of TLSTM, this research propose to utilize TLSTM based network to estimate the change in data characteristics over time. Further, to mitigate the problem of generating labeled data under supervised framework, this research consider an unsupervised framework and use the regeneration error as the criteria for detecting attacks. The idea is, *Given a model trained on a normal traffic to regenerate the input data, the regeneration error for attack data should be higher*. Therefore, a machine learning-based method is required that utilize normal network traffic data to train with. In practice, it is easy to obtain the normal traffic flow data by using various utilities like Netflow [106], CICFlowMeter [102], etc. These normal flow records can be used to learn the characteristics of normal traffic and with the learned parameters, an attack can be identified based on estimated error.

The proposed method works by training the TLSTM autoencoder model using extracted contextual features on normal flow records with low reconstruction error and in case of attack, the trained model produces a larger reconstruction error to trigger an anomaly. An Autoencoder [107] is one of the state of the art deep learning based method which is used for unsupervised anomaly detection [108–110]. It pertains to profiling the normal network traffic, and identifying any deviation from the normal as an anomaly. It consists of two parts Encoder and Decoder. With the given input of N dimensional vector, encoder compresses it into latent vector represented as k where $N > k$ and decoder thereafter reconstructs the compressed latent vector back to N dimension. An Autoencoder trained with Normal traffic learns the pattern for reconstruction for normal data and classify input instance as attack if reconstruction error is larger than predefined threshold.

3.1.2 Contributions

The main contributions of this chapter can be summarized as follows:

- Time aware LSTM based autoencoder is utilized to capture the influence of temporal context and regularize the features using the TLSTM based autoencoder network.
- The proposed system is based on unsupervised setup to estimate re-construction error, and identify attacks based on a threshold obtained only from normal traffic.

- This research work utilize multiple recent benchmark datasets - CICIDS2018, and CICDDoS2019 for evaluating the performance of the suggested method for FTP BruteForce (FTPBrute) and UDP Denial of Service attacks (UDPDoS).

The chapter is organized in different sections as follows. Related works are discussed in Section 3.2. Section 3.3 explains the proposed method in detail and the components involved in the processing. The implementation of the proposed approach and experimentation results are presented and a comparison with existing solutions is discussed in Section 3.4. In Section 3.5 the summary is presented.

3.2 Related Works

Deep learning models have been predominantly used recently in the area of network intrusion detection as they can learn the discriminatory features automatically and capture the non-linearity in a high dimensional dataset. Various deep learning based solutions have been proposed earlier to capture the spatial and temporal context, mentioned as follows.

3.2.1 Supervised learning based methods

Supervised learning-based solutions like HCRNN [50] is proposed earlier in which the author has used the concept of utilizing the spatial and temporal features together using convolution neural network model (CNN) and recurrent neural network (RNN) layers in sequence. It initially processes the features through the proposed CNN network and later passed through the RNN network to generate the sequence at each timestep. CNN is mainly used to capture the local features while temporal features are captured using RNN. To handle the data imbalance issue, oversampling strategy is utilized and the proposed method achieved 97.75% of accuracy with 10-fold cross-validation on CSE-CIC-IDS2018 dataset.

Another online supervised learning-based method using LSTM for abnormal traffic detection is proposed in DeepWindow [111]. In the proposed method LSTM is used to capture the previous contextual information and MIMC(Mutual Information and Maximal Information) coefficient method is used for feature selection. Features like count packets, count PSH flag, total time, packet max, average length, etc. are computed for a fixed time window on the CICIDS2017 dataset.

Being a supervised solution, it desires the data to be revised periodically to detect the

new attack. The major limitation of these proposed solutions is that it is unable to handle the irregular time interval between events and assign equal weights to previous events.

3.2.2 Hybrid learning based methods

Anomaly can also be detected by utilizing the concept of regeneration error through autoencoders. LSTM-based autoencoder along with One class SVM is proposed in [28] to detect the anomalous behavior using reconstruction error threshold on InSDN dataset [112] specifically for attacks in SDN (Software defined networking) environment. It uses a context length of 1 which is not taking into consideration of the previous context and lacks in capturing the effect in subsequent flows. Another method to capture the spatial context [113] is proposed, which is using a convolutional autoencoder with one class SVM. In this method, multiple convolutional layers (CNN) are used for representation learning and the joint optimization framework by defining reconstruction and classification error into a unified objective function is proposed in this scheme. Another approach based on an ensemble of autoencoder-based detection systems is proposed in Kitsune [27] that is used to catch attacks without supervision. It is based on training autoencoders on normal traffic and performs anomaly detection based on estimated reconstruction error. The proposed method computes 115 statistical features over 5 last time windows and applies a dampening effect over time. The method consists of two parts an ensemble layer for training and the output Layer for producing the final score of anomaly. To reduce the number of features agglomerative hierarchical clustering is primarily used.

Authors in [114] proposed a sparse autoencoder-based model combined with a kernel function. The overall optimization in this proposed method is performed using a genetic algorithm and the method is tested on botnet detection data. DeepStream [115] proposed an autoencoder-based stream clustering method and identify the anomaly based on distance and reconstruction error method. Authors in [116] suggested a two-stage autoencoder model for anomaly detection. Autoencoder in the first stage is used to filter high probability normal traffic and variational autoencoder(VAE) in the second stage is used to identify attacks using reconstruction probability. Due to the introduction of VAE, the proposed solution suffers from high computation time. Most of these hybrid models suffer from high computation time and complexity.

3.2.3 Similarity Learning based methods

Attacks can also be detected by comparing the learned representations of normal, attack-specific scenarios individually and comparing the distance between the pairs. It is expected that distance is higher in case of anomaly presence. This concept is used to train the special networks like triplet networks proposed by authors in [117]. In this method autoencoder-based metric learning for network intrusion detection by combining autoencoders with triplet networks is proposed. It requires two separate autoencoders to be trained on normal and attack data. The learned triplet network is used to separate attacks from normal traffic. Another paper [118] propose training the siamese network to differentiate between class based on pair similarities using constructive loss. The proposed methods require data segregated into normal and attack classes and it is also unable to capture the temporal context.

In this section, recent research work is studied where autoencoder-based deep learning models have been used predominantly for detecting various attacks. Some of the proposed solutions consider the previous context but do not capture the irregular time interval and does not use the decay function to reduce the effect of prior values with time. With this motivation, the proposed method overcomes the existing limitations of each method as specified in table 3.1. The proposed methodology is discussed in detail in the next section 3.3.

3.3 Proposed Methodology

Data regeneration is a well-known unsupervised method used to detect anomalies in given network traffic records. This re-generation of data can be performed using autoencoder based models. The proposed method is based on estimating this regeneration error through autoencoder-based network. After training the proposed network, the threshold θ is determined which is used at the inference stage to detect if the input traffic is Benign or Attack, an error is estimated using a learned autoencoder and compared with the learned threshold values. It is expected that for Benign traffic low regeneration error occurs while in the case of Attack traffic high regeneration error is estimated through the autoencoder network.

To consider the contextual information from the sequence of traffic records specific context window of size Δw is pre-selected to process the records. This window represents how much historical information is to be taken into account to predict the current state. The pro-

Table 3.1: *Comparison of various available methods*

Method	Labeled Data Requirement	Processing Stages	Complexity	Diminishing Context
Supervised Learning based methods [3.2.1]	Labeled Data is required to build models.	Single classification model	Low complexity to build model.	Not considered.
Hybrid Learning based methods [3.2.2]	Labeled data is not required.	Multiple models are trained.	High complexity to train models.	Not considered.
Similarity Learning based methods [3.2.3]	Required data separated for Normal and Attack traffic.	Separate models are trained for each type of data.	High complexity to train models.	Not considered.
Proposed Method [3.3]	Labeled data is not required. Only normal traffic is considered.	Single model trained only on Benign data.	Low complexity to train models.	Diminishing context is considered.

posed method considers the flow level records, available in the dataset as specified in section 3.4.1, for training the model. These records consist of bi-directional network flow-related statistical information like total packets, total size, packet rate, byte rate, inter-arrival time, TCP flag information, etc. that are pre-calculated by flow analysis tools. At the initial stage, these feature values are normalized on a common scale explained later in section 3.3.1. To capture the temporal context from the incoming flow of records that are ordered by timestamp, a context of fixed length Δw is used to capture the temporal propagation of information. Context length determines how much of the previous context needs to be captured in processing. Figure 3.1 shows the schematic diagram of the proposed system.

Let X be the sample input data with context of length n. Then, it consists of records x_1, x_2, x_3, x_4 , . . , x_n generated at subsequent timestamps t_1, t_2, t_3, t_4 , . . t_n . Each of the records (represented with normalized features) is supplied to TLSTM Encoder (discussed at section 3.3.2). The encoder module consists of multiple layers and each layer is associated with multiple TLSTM units. The flow of records is processed through each TLSTM unit where the output of the next unit is based on the previous context. At the last stage of the encoder, intermediate representation is generated which is provided to the TLSTM decoder module. The decoder module consists of the same number of stages as specified in corresponding encoder layer and tries to re-construct the supplied information at the

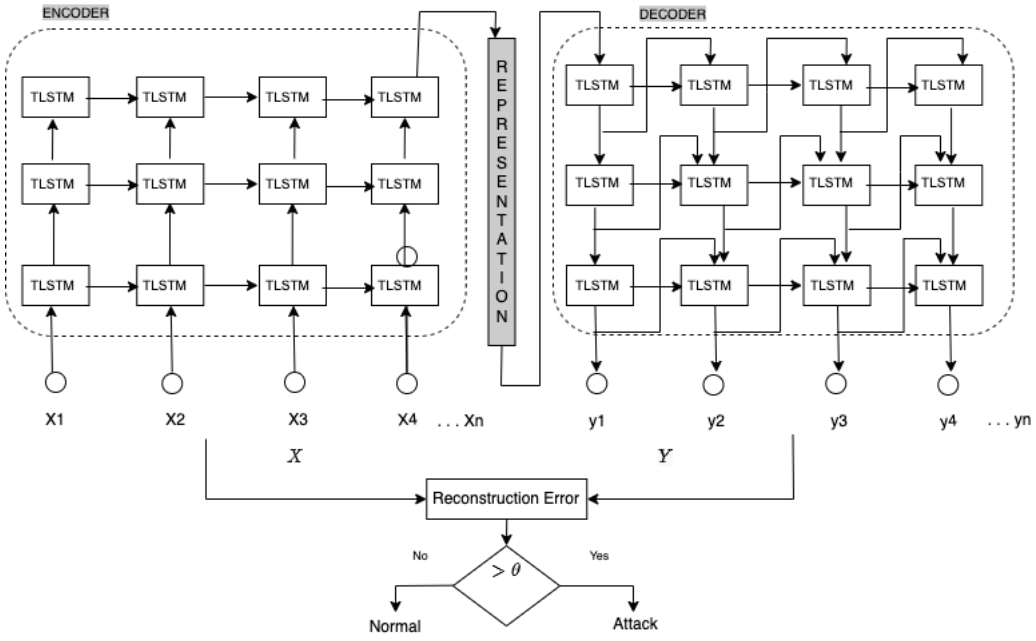

Figure 3.1: *Processing diagram for T-LSTM based Autoencoder Network*

subsequent stage of processing. At the final stage, decoder generates the output as Y which is of the same dimension as the input X. Now the reconstruction error based on provided loss metric is estimated as defined in section 3.3.2, which helps us to identify the threshold θ for normal traffic. Once the threshold is determined, incoming flow is identified to be an attack specifically if the reconstruction error is larger than threshold θ. All the components are executed in succession for every incoming flow in Δw context of flow records to identify attack activity. The proposed solution consists of *Data Preparation* and *Model Building* as major components as described below:

3.3.1 Data Preparation

Data Preparation is a prerequisite for any modeling task. The available flow level numerical features as specified in tables 3.2 and 3.3 are utilized. These feature set consists of various flow level numerical statistics like packet transfer rate, length, inter-arrival times, sub-flow information, header statistics etc. Since the feature values follow different ranges, so to avoid the model bias, it is required to transform them to a common scale. For this purpose, Normalization is used to transform features in with mean and standard deviation as $(0, 1)$

respectively. For feature values shown as X, the transformed value is represented as X_t, mean and standard deviation of X are μ and σ respectively. Equation 3.1 represents the normalization process.

$$X_t = \frac{X - \mu}{\sigma} \tag{3.1}$$

After normalizing the dataset, It is required to supply the sequential information for modeling. The provided context length Δw is used to process the sequence of flow records. The specified context length determines how much the previous context is utilized to learn the temporal information.

Table 3.2: *Feature set Information for CICIDS2018 [1] dataset*

S.No.	Feature Category	Feature Description	Count
1	Packet Transfer rate	Features specific to forward and backward packet transfer rate	5
2	Packet Length	Features specific to forward and backward packet length information	15
3	Inter-arrival Time	Elapsed time information between flows	14
4	Header information	Features specific to header information like flag count, header length etc.	14
5	Segment information	Features specific to segment size information	4
6	Transfer rate	Features specific to bulk transfer rate	7
7	Subflow related	Features specific to sublow information	7
8	Flow activeness	Statistical Features specific to flow activeness information	8

Table 3.3: *Feature set Information for CICDDoS2019 [2] dataset*

S.No.	Feature Category	Feature Description	Count
1	Packet Transfer rate	Features specific to forward and backward flow packet transfer rate	5
2	Packet Length	Features specific to forward and backward flow packet length information	16
3	Inter-arrival Time	Elapsed time information between flows	14
4	Header information	Features specific to header information like flag count, header length etc.	15
5	Segment information	Features specific to segment size information	2
6	Transfer rate	Features specific to bulk transfer rate	7
7	Subflow related	Features specific to sublow information	4

3.3.2 Model Building

3.3.2.1 Time aware LSTM Model

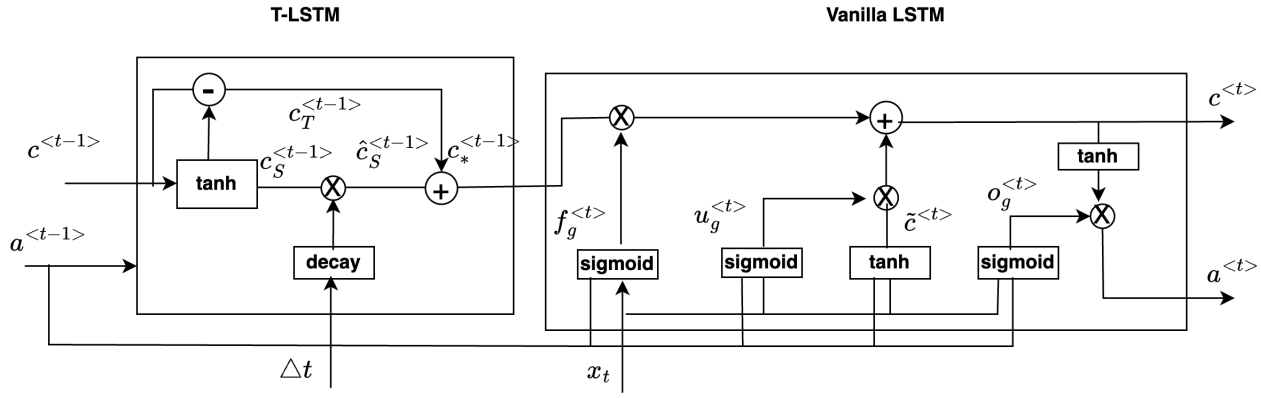

Figure 3.2: *Time Aware LSTM*

TLSTM [103] model contains long short term memory unit capable of handling irregular time intervals in data. It uses elapsed time information between consecutive events and adjust cell memory inherently. As shown in figure 3.2, Time aware LSTM model unit consists of two parts Time aware unit (T-LSTM) and Vanilla LSTM unit. Let us assume for given sequence of inputs x_1, x_2, ... x_t, model computes sequence of output as y_1 ,y_2. . . .y_t. Let $W_f, W_u, W_o, W_a, W_d, W_c$ and $b_f, b_u, b_o, b_a, b_d, b_c$ are trainable parameters, $C^{<t-1>}$ and $C^{<t>}$ represents previous and next cell states respectively. As represented from

the recurrence in equations (3.2), Time-aware units compute cell states $C_*^{<t-1>}$ using the specified decay function with $\triangle t$ as time interval. This decay functions determines the amount of information decay with time.

$$\begin{cases} \mathrm{C}_S^{<t-1>} = tanh\ (\ W_d\ C^{<t-1>} + b_d) \\ \hat{C}_S^{<t-1>} = C_S^{<t-1>} * decay(\triangle t) \\ C_T^{<t-1>} = C^{<t-1>} - C_S^{<t-1>} \\ C_*^{<t-1>} = C_T^{<t-1>} + \hat{C}_S^{<t-1>} \\ \tilde{C}^{<t>} = \mathrm{tanh}(\mathrm{W}_c\ x_t + U_c\ a^{<t-1>} + b_c) \end{cases} \tag{3.2}$$

The modified cell state is $C_*^{<t-1>}$ is finally used in Vanilla LSTM part to compute the forget $f_g^{<t>}$, update $u_g^{<t>}$ and output gate $o_g^{<t>}$ states and ultimately the final cell state $C^{<t>}$ and activation values $a^{<t>}$ are computed as shown in equations (3.3).

$$\begin{cases} \mathrm{f}_g^{<t>} = sigmoid\ (\ W_f\ [\ a^{<t-1>}\ ,\ x_t\ ] + b_f) \\ \mathrm{u}_g^{<t>} = sigmoid\ (\ W_u\ [\ a^{<t-1>}\ ,\ x_t\ ] + b_u) \\ \mathrm{o}_g^{<t>} = sigmoid\ (\ W_o\ [\ a^{<t-1>}\ ,\ x_t\ ] + b_o) \\ \mathrm{C}^{<t>} = f_g^{<t>} * C_*^{<t-1>} + u_g^{<t>} * \tilde{C}^{<t>} \\ \mathrm{a}^{<t>} = o_g^{<t>} * \mathrm{tanh}(\mathrm{C}^{<t>}) \end{cases} \tag{3.3}$$

3.3.2.2 TLSTM Autoencoder Model Training

The main objective of this stage is to train the TLSTM based autoencoder model. The pre-defined context length (To use the information from previous timesteps) is selected to feed into TLSTM based model that is used to define the number of units in the input layer. The proposed model consists of encoder and decoder stages and each stage consists of multiple TLSTM units as shown in figure 3.1. Encoder block reduces the dimensions to an intermediate representation that illustrates the compressed input data. The compressed format is then fed into decoder blocks for generating the output feature vector and layers in decoder blocks are arranged in reverse order as encoder layers. The final layer of the decoder generates an output vector and Mean Square Error (MSE) is used to calculate the error between input and output vector using equation (3.4). Let X be input sequence with n feature vectors corresponding to N records and Y be the corresponding output vector

sequence then mean squared error MSE_X can be estimated using equation (3.4).

$$MSE_X = \frac{1}{n}\sum_{i=1}^{n}|X_i - Y_i|^2 \tag{3.4}$$

The estimated error helps us to identify the reconstruction error threshold θ, which is based on the mean of the error values measured for all N records in normal traffic as shown in equation (3.5). It is expected that reconstruction error for normal traffic data can be less as compared to that of anomalous traffic data. This behavior help us in detecting anomalous traffic if it is greater than the defined threshold.

$$\theta = \frac{1}{N}\sum_{x=1}^{N} MSE_X \tag{3.5}$$

The total complexity of the training an autoencoder model for two layers of size N is O(N^2) where N is the number of neurons in each layer. Let us assume there are k layers in autoencoder network then the overall complexity is O(k * N^2). Since k <<<N, therefore complexity can be assumed as O(N^2)

3.4 Experimentation Results

3.4.1 DataSet

Two different publicly available network attack-specific datasets are considered to evaluate the performance of the proposed method. The features of the datasets are shown in the Table 3.4

Table 3.4: *Information on Dataset parameters used for experimentation*

Dataset	**Attack**	**Features**	**Class**	**Data length**
CICDDoS2019	UDPDoS	63	Normal	30109
			Attack	3134645
CICIDS2018	FTP BruteForce	74	Normal	667626
			Attack	380949

3.4.2 Baseline Model

Following two baseline unsupervised models are considered for comparison.

- Multi-layer Perceptron (MLP): Instead of TLSTM unit, MLP units are considered in the proposed auto-encoder framework with the same deep layer.
- LSTM: Instead of TLSTM unit, LSTM units are considered in the proposed auto-encoder framework with the same deep layer.

3.4.3 Implementation

The flow level data for normal traffic captured for a specific dataset is utilized and all the available features are selected to train the TLSTM Autoencoder model. The proposed scheme is implemented on machine with Mac OS consisting of intel core i7 processor with 16 GB RAM and TensorFlow python package is used for the implementation. The following configuration for an autoencoder network is used as shown in Table 3.5. The same configuration has been used for the baseline models as well.

Table 3.5: *Model Configuration. The* **f** *indicates input feature vector.*

Parameter	**Values**
Total Layers	4
Encoder structure	**f**,128, 64, 32
Compressed output length	16
Decoder structure	32, 64, 128, **f**
Optimizer	Adam
Learning rate	0.0001
Loss function	Mean squared error
Batch size	32
Number of Epochs	20

3.4.4 Results and Analysis

In this section, the detection performance of the proposed method is estimated for varying context length. The TLSTM and LSTM processing units are utilized respectively with varying context length (upto 20) and evaluated the performance with metrics like Precision, Recall, F1-score, and Accuracy as shown in Table 3.6. To estimate the prediction

performance, 10% of normal data is hold out as testing along with the attack dataset, and estimate the evaluation metrics between *Benign* or *Attack* as classes. From the results, it is evident that higher context length provides better overall classification metrics at a certain point. This information can help us to select the optimum value for the context length. Due to data imbalance, it is analyzed that Recall is very high in both TLSTM and LSTM as there are fewer overall false negatives.

It is also evident that TLSTM is performing better than LSTM overall in terms of provided metrics. The best metric values is specified in bold that is obtained in various experiments for different datasets. It is analyzed that percentage of improvement in CICIDS2018 dataset is around 3.2% in F1-score and 5.5% in Accuracy respectively for TLSTM model but the amount of metrics improvement in CICDDos2019 dataset through TLSTM based model is not very high (0.3% in F1-score and 1.2% in accuracy), that may be due to less number of training samples present (Table 3.4) in the dataset and time decay factor is not able to generalize the attack sequence well.

It is also analyzed that TLSTM model provides high variation in metrics over different context length. However, the LSTM model provides a very small variations for different context length, for all dataset. The estimated standard deviation on F1-score and Accuracy for both the datasets for TLSTM based model is (for CICDDoS2019 (1.3, 2.6) & for CICIDS2018 (1.5, 2.3)), and for LSTM based model is (for CICDDoS2019 (0.3, 0.6) & for CICIDS2018 (0.5, 0.5)) respectively.

Table 3.6: *Context Length vs Detection Performance*

Dataset	Context Length	Precision%		Recall%		F1-score%		Accuracy%	
		LSTM	**TLSTM**	**LSTM**	**TLSTM**	**LSTM**	**TLSTM**	**LSTM**	**TLSTM**
CICDDoS2019	2	94.6	88.0	100	100	97.2	94.1	95.0	89.2
	4	94.6	90.8	100	100	97.2	95.1	95.0	91.2
	6	93.6	90.1	100	100	96.7	94.7	94.0	90.4
	8	93.9	90.8	100	100	96.9	95.0	94.3	90.8
	10	94.3	95.1	100	100	97.1	97.5	94.7	95.6
	12	93.0	95.0	100	100	96.3	97.4	93.4	96.0
	14	94.7	94.4	100	100	97.2	97.1	95.1	95.4
	16	94.8	**95.3**	100	100	97.3	**97.6**	95.3	96.2
	18	94.8	95.0	100	100	97.3	97.6	95.3	**96.2**
	20	94.9	94.6	**100**	**100**	97.3	97.2	95.3	95.6
CICIDS2018	2	88.6	90.4	100	100	93.9	94.5	90.4	92.0
	4	89.9	89.9	100	100	94.7	94.8	91.6	91.7
	6	89.8	90.0	100	100	94.6	94.8	91.6	91.7
	8	88.6	91.3	100	100	93.9	95.4	90.4	92.8
	10	89.6	90.7	100	100	94.5	95.2	91.3	92.3
	12	88.8	96.1	100	100	94.4	98.0	90.5	96.9
	14	89.0	94.1	100	100	**95.5**	95.2	91.1	92.6
	16	89.2	96.3	100	100	95.4	98.1	91.2	97.1
	18	90.3	96.1	100	100	94.9	98.0	**91.9**	96.9
	20	89.0	**96.7**	**100**	**100**	95.2	**98.3**	91.1	**97.5**

In the previous experiment [119, 120] anomaly has been detected using regeneration error which describes that One class SVM (OCSVM) is another well known method to detect network anomalies. It has been analyzed earlier by [28] that utilizing learned representation from the auto-encoder and providing it to One class SVM (OCSVM) model improves the overall attack detection performance. This motivates us to utilize the learned intermediate representation from the auto-encoder (taking encoder output only) and use an outlier detection algorithm i.e OCSVM to find the anomalous points. The results are estimated in table 3.7. For comparison, the model is selected which is performing the best in terms of F1-score and Accuracy from table 3.6. It is estimated that the performance of the standalone OCSVM-based model is improved further after applying proposed model on learned representations using LSTM and TLSTM models. It can also be seen that the TLSTM based OCSVM model is performing better than LSTM based OCSVM model for all the data sets.

Table 3.7: *Performance Comparison for OCSVM based models.*

Dataset	Method	Context Length	Precision%	Recall%	F1-score%	Accuracy%
CICDDoS2019	OCSVM	-	84.89	100	87.3	87.7
	LSTM + OCSVM	20	85.8	100	92.3	85.8
	TLSTM + OCSVM	16	**87.7**	100	**93.4**	**87.9**
CICIDS2018	OCSVM	-	50.0	90.1	64.3	50.0
	LSTM + OCSVM	18	62.1	100	76.6	63.4
	TLSTM + OCSVM	20	**65.2**	**100**	**78.6**	**65.2**

From the tables 3.6 and 3.7, it is evident that the maximum performance of both the autoencoder models (i.e. LSTM and TLSTM) is achieved at larger context length as more information is captured. For LSTM based model best f1-score is attained at context length (20, 14), and for TLSTM based model best f1-score is retreived at (18,20) respectively for CICDDoS2019 and CICIDS2018 dataset.

Further, this section also investigates the number of flows per second processed by the trained model against the context length (upto 20), as shown in figure 3.3. The figure shows that the flow processing rate is reduced by taking a larger context length which happens due to higher processing overhead. With the proposed results, the optimum context length can be selected to process the data in an estimated time.

3.4.4.1 Time Interval vs Performance

Time interval $\triangle t$ (as seen in equation (3.2)) plays a vital role in the TLSTM model. It is used to specify the interval across previous contexts that decides how much decay factor is introduced on previous time values. Time interval is specific to attack as each attack has different relation to previous context. To decide appropriate time interval for a given context length, the performance metrics at various values of time intervals is analyzed. The fixed context length of 10 is considered, and the performance metrics are estimated with varied time intervals. The best suited time interval $\triangle t$ for each dataset is used for the other experiments with different context length. Though, there is a possibility that for different context length the best possible time interval is also slightly different. However, to set the value uniform over different setup, the context length 10 has been considered. Figure 3.4 shows the performance of TLSTM with context length 10 against different time interval. Recall is not considered into comparison as it is high in most of the scenarios due to data imbalance. It is evident from the figure that for ftp brute force attacks in CICIDS2018 dataset the best performance is obtained at 6. For the UDPDoS attacks in CICDDoS2019,

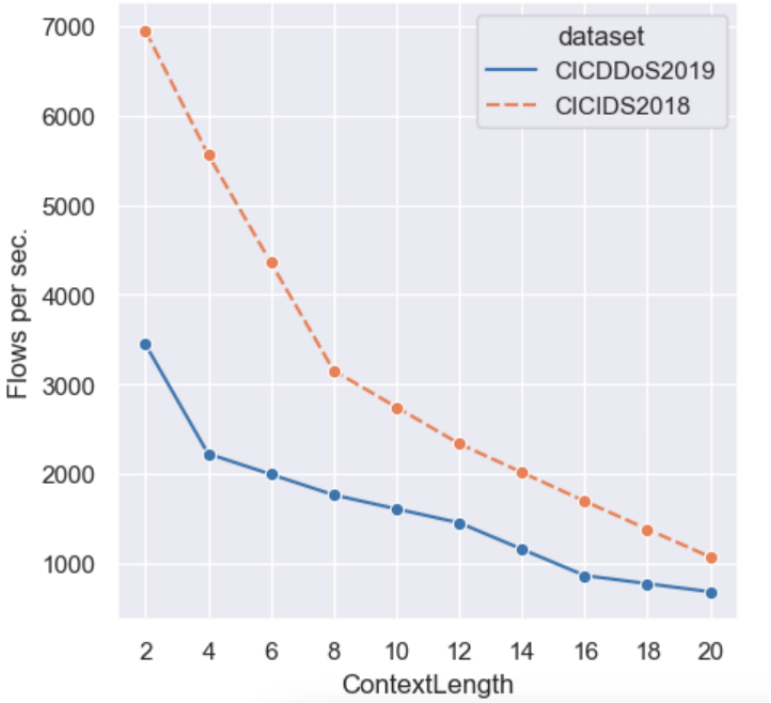

Figure 3.3: *Context Length vs Flows per sec.*

this value is identified as 10.

3.4.4.2 Baseline Comparison

In this section the various autoencoder models build with different inherent units are considered. The performance is estimated for MLP based Autoencoder, LSTM based autoencoder, and proposed TLSTM autoencoder for comparison. The same model configuration as defined in table 3.5 is used. Only the inherent computation units like MLP, LSTM, and TLSTM are changed for the individual model. The experimentation is performed with different context lengths upto 20 records. Different context length reports different performance. Table 3.8 shows the performance of model for the best context length. The specified context length has been selected from table 3.6 for which the highest F1-score and Accuracy are obtained. It shows the proposed model outperforms both the MLP and LSTM auto-encoder for all the datasets. In Table 3.8 the best-performing results are compared for individual MLP, LSTM, and TLSTM based models on specific datasets and it is estimated that TLSTM based model performs the best in all the scenarios.

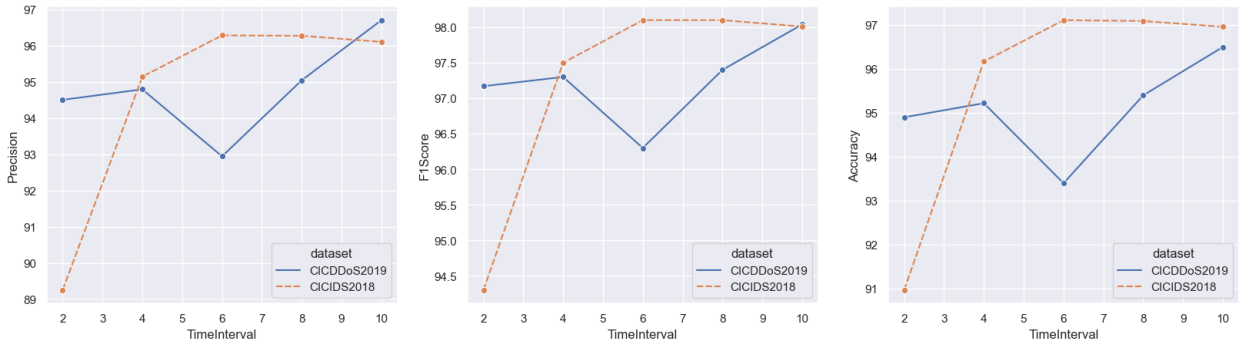

Figure 3.4: *Performance of the proposed model over different Time Interval for different datasets*

Table 3.8: *Performance Comparison of different auto-encoder models.*

Dataset	Method	Context Length	Precision%	Recall%	F1-score%	Accuracy%
CICDDoS2019	MLP	-	93.6	100	96.7	94.3
	LSTM	20	94.9	100	97.3	95.3
	TLSTM	16	**95.3**	100	**97.6**	**96.2**
CICIDS2018	MLP	-	89.0	100	94.2	90.7
	LSTM	18	90.3	100	94.9	91.9
	TLSTM	20	**96.7**	100	**98.3**	**97.5**

3.5 Summary

This research work presents a method for attack detection that requires only Normal traffic records for training. With the use of the proposed method, labeling every network flow data for a specific attack can be avoided. The latest and representative flow-level dataset is used for the implementation. The proposed approach is capable of detecting whether an attack is present or not based on reconstruction error from a trained autoencoder model. It also tested the performance of different processing units like MLP, LSTM, and TLSTM in the same auto-encoder architecture. From various experiments on CICDDoS2019, and CICIDS2018 datasets, it is evident that the proposed model outperforms its baseline counterparts for various attack types like DoS, DDoS, and BruteForce attacks. The results for various time interval values specific to each attack type are analyzed. This research work also analyzes the performance with outlier detection-based OC-SVM models and utilizes the learned representation from the autoencoder based model into OC-SVM based model to identify the attack. The next chapter extend this work with protocol specific introduction

systems.

4

Protocol Aware Network Intrusion Detection System

4.1 Introduction

The previous chapter discusses the influence of temporal context in an unsupervised learning-based network intrusion detection system. Various processing units like MLP, LSTM, and TLSTM are used to build the autoencoder-based model that is based on an unsupervised mechanism and can be easily implemented using normal network traffic. Since the behavior of attacks change with the specific protocol behavior, therefore for better discrimination and detection of attacks it is important to consider the protocol information while attack detection and enabling the model to follow protocol-aware decisions. In this direction, this research work proposed a method of utilizing the protocol-specific information while determining the attacks and developing a method that is protocol-aware in nature.

4.1.1 Need for Protocol aware IDS

In TCP/IP protocol suite [57], various protocol exists at different layers of networking that are associated to specific application like SSH(Secure Socket Shell), FTP(File Transfer Protocol) , HTTP(Hypertext Transfer Protocol) etc. These protocols are inherently vulnerable

to different types of attacks like Brute Force, Denial of Service (DoS), Distributed Denial of service (DDoS) etc. Attackers use various available open-source utilities like LOIC [5], Slowloris [6], Nmap [7], etc. to perform different types of attacks easily. These attacks are performed by exploiting existing vulnerabilities in these networking protocols.

Over the years IDS systems have been developed using machine learning techniques [14, 16, 45] that consider building a single machine learning model for a group of attacks associated with different protocols, however, attack characteristics may be different for different protocols. Further, for some protocols data is in abundance, and for some protocols very few records exist that can introduce bias for the trained model. Therefore, it is important to develop an IDS which is protocol aware and learn the representation of each attack individually. Further, these representations can be utilized to identify attack activity using protocol-specific IDS and learn the importance of each protocol channel.

Most of the existing methods have considered protocol specific attack detection in supervised [46] or signature [47] manner. Due to fixed signature, the earlier proposed methods require labeled data to train with, and fixed signatures are specific to attack type, thus failing to detect zero-day attacks. And, it also requires huge manual effort for label generation, and needs data/signature updates over time to incorporate new attacks. This motivates us to develop an unsupervised machine learning-based method that needs to be protocol aware and requires only normal data to train with. Since, it is easy to obtain the normal traffic flow data by using various utilities like Netflow [106], CICFlowMeter [102], etc. and using these normal flow records, normal traffic characteristics can easily be learned. The proposed method identify the attack activity if high deviation from the learned normal characteristics is identified.

Protocol-aware IDS can also provide the additional benefits mentioned as follows.

- **Better scalability and load balancing:** With protocol-aware IDS resource exhaustion can be avoided by running detection for specific attacks only. On the other hand for large network, dedicated servers can be easily deployed that may increase system's scalability for high speed networks.
- **Data imbalance handling:** Network traffic distribution is not even for every protocol therefore amount of data in some protocols is high in volume while others contain few records. With the protocol aware method model bias towards attack with the high volume of available data can be avoided.

- **Capture network specific attacks:** Without a profile-based solution, detection system looks for all kinds of attacks on every network packet or group of packets, irrespective of the fact that some networks never launch them.

Motivated by above mentioned reasons, this research propose the protocol-aware method for intrusion detection that is based on an unsupervised learning mechanism and can work without labeled data-set, and is further capable for detecting unseen attacks.

4.1.2 Idea & Contributions

In earlier unsupervised learning-based methods [23, 24], attack detection is proposed by training the model with normal traffic and learning the characteristics of benign traffic. It employs attack detection in case of a huge deviation from the learned hypothesis. Autoencoder-based methods are primarily used for this purpose which can be used to learn the normal traffic characteristics and utilize regeneration error as a measure to estimate the high deviation in case of attack occurrence. This research consider protocol-aware autoencoder framework and employ this concept of regeneration error for attack detection. From various experiments over CICIDS2018 [1] and CICDDoS2019 [2] datasets, it is evident that the proposed protocol-aware model outperforms its baseline counterparts for various attack types like DoS, DDoS, and BruteForce attacks. The main contributions of this chapter can be summarized as follows:

- Protocol specific unsupervised autoencoder based model is developed to capture time dependent contextual information and identify specific attacks at protocol level.
- A method to combine protocol specific encoders with attention [95] network to identify importance for each protocol channel for attack identification is also proposed.
- The proposed system is based on unsupervised setup to estimate re-construction error, and identify attacks based on a threshold obtained from normal traffic.
- Multiple recent benchmark datasets - CICIDS2018 [1] and CICDDoS2019 [2] are utilized for evaluating the performance of the proposed model on various attacks.

The rest of the chapter is organized as follows. In section 4.2 various solutions for protocol-specific methods are discussed. Section 4.3 presents the approach for building protocol protocol-aware IDS solution. In section 4.4.1 the implementation of the proposed

scheme is detailed and experimentation results are presented. Finally in section 4.5 the summary is discussed.

4.2 Related Works

4.2.1 Supervised learning based methods

Most of the existing solutions are built on protocol-specific model development which consider applying filter at the protocol, host, or virtual machine level. Most of these solutions first apply protocol-specific filters for data reduction and further train the supervised models on the reduced dataset. These kinds of solutions lack protocol awareness while taking the decision on attack presence and building the model on protocol-specific datasets alone. For example, a supervised learning-based solution proposed in this area by Raman [46] includes alpha Profiling (protocol level) for time complexity reduction and beta Profiling (Clustering similar profile and pick center) for data size reduction. The proposed detection mechanism achieves 97.67 % of accuracy with 1.74% of false positive rate for the multi-class NSL-KDD dataset. Later profile-based IDS solution is proposed [47] that works at Virtual Machine (VM) level where network traffic is filtered based on VM's IP address and a VM profile is created which is matched from existing attack signatures for the attack activity. Another supervised learning-based Protocol specific IDS is proposed [48] for DoS and DDoS attacks that work by extracting protocol-specific features and training classifiers at the packet level. In this scheme incoming traffic is separated based on protocols like TCP, UDP, and ICMP, further protocol-specific DoS/DDoS features are estimated for classification.

The concept of high deviation of the current profile from the learned normal profile has also been used for attack detection. Like, the solution proposed by [121] dynamically and actively profiles, and monitors all networked devices for the detection of IoT attacks, and any deviation from the defined profile is considered to be an attack. The proposed solution consists of two main parts i.e. Network Profiling component and IDS Component. The network profiling component uses rate-informed heuristic profiling to create an expected throughput pattern for each device on the LAN and generates hourly, daily & weekly profiles per device. A percentage difference is calculated, comparing the rate profile of the new capture to each timed profile and if this is above the threshold, an anomaly is detected. The mentioned IDS Component is a machine learning module that makes use of the Hilbert space-filling curve as its main clustering algorithm to convert the bytes into a 2D image and

train the image classification model using the MobileNetV3 [122]. The combination of both systems results in an accuracy of 99.17%.

The major limitation of these proposed methods is that it is supervised in nature and build models on filtered datasets that ignore protocol awareness.

4.2.2 Unsupervised learning based methods

On the other hand, autoencoder-based models have also been primarily used for attack detection using unsupervised mechanism, that is based on estimating re-generation error, but lacks protocol awareness for decisions. For instance, some of the autoencoder-based solutions proposed by [27, 28, 113, 114] and a few shot learning-based methods utilizing autoencoder representation, proposed by [117, 118] are used earlier for the group of attacks and are not protocol aware (does not consider protocol specific information) and most of these models suffers from high computation time and complexity. The method proposed by [123] for IIoT networks is based on distributed AI and is implemented using multiple autoencoders trained for each local processing unit and computations are combined using the Adaboost model for the final conclusion. The main limitation of this proposed model is that it does not consider temporal context inherently. Another unsupervised solution proposed in [124] works by training a different ensemble of neural networks on protocol header-specific information and identifying anomalies based on error threshold. In this proposed method each protocol is converted to a set of normalized numeric features and processed by 5 different neural networks: deep autoencoders, deep MLPs, LSTMs, BiLSTMs (Bi-directional LSTM), and GANs (Generative Adversarial Networks), and the final anomaly score is computed based on anomaly scores of other protocols which is computationally complex.

The proposed model resolves these issues by introducing a protocol-aware IDS solution that is based on an unsupervised learning mechanism, is easy to build, and produces results with lesser complexity. In the next section, the proposed methodology is discussed in detail.

4.3 Proposed Methodology

Given a network traffic data, a commonly used unsupervised approach to detect anomalies in the data is to regenerate the data using the method like auto-encoder. An autoencoder is an encoder-decoder model where the encoder takes an input sample and produces an intermediate representation of the input sample. The decoder takes the intermediate rep-

resentation and attempts to regenerate the input sample again. In short, given an input sample x, autoencoder regenerates the input $\acute{x}$ again, and can be defined as follows.

$$\acute{x} = decoder(encoder(x)) \tag{4.1}$$

where $\acute{x}$ is the regenerated output of x. For anomaly detection task, the autoecoder is generally trained using normal data, where the regeneration error (say, $e = (x - \acute{x})^2$) is expected to be low for normal data, and high for anomalous data.

Majority of the earlier studies of detecting network attack using autoencoder [27,28,113] consider entire cross-sectional network data (i.e., entire volume of network data received by a device at a time instance or a window of time range). By considering the entire data, encoder part of the autoencoder generates a global representation which summarizes the input data into a latent space. The data received by a device at a particular instance in time may consist of *heterogeneous information* related to different protocols and anomaly characteristics of different protocol specific attacks may also be different. Capturing differences in characteristics of different protocols may be important to effectively determine anomalies present in data. However, a generalized global representation may fail to effectively capture heterogeneous characteristics of different protocols, which may be important for anomaly detection effectively. Motivated by the above concerns, the proposed method not only generates global representation of the entire data, but also generates local representation for different protocols. If we apply autoencoder only to protocol specific data, then we refer to this specific representation as *protocol specific representation*.

Since incoming traffic can contain specific or multiple protocol related information at a time, the decision of attack requires to be protocol aware (specific input protocol channel should impart higher weightage in decision) and needs to be independent of input type (weightage needs to be learnt automatically). In case incoming traffic belongs to particular protocol, the protocol specific representation requires to represent the information better (fine representation) and on the other hand if the traffic does not belong to that specific representation, then it should produce coarse representation.

Figure 4.1 presents the diagram of model training for the proposed protocol aware scheme. It utilizes training data that is extracted from the flow-level information of network packets. The training data is pre-processed in the first step by applying normalization on numerical features. In the next stage, the fine representations are learned by training the protocol-specific auto-encoders on the protocol-specific dataset (FDS_n), and the global

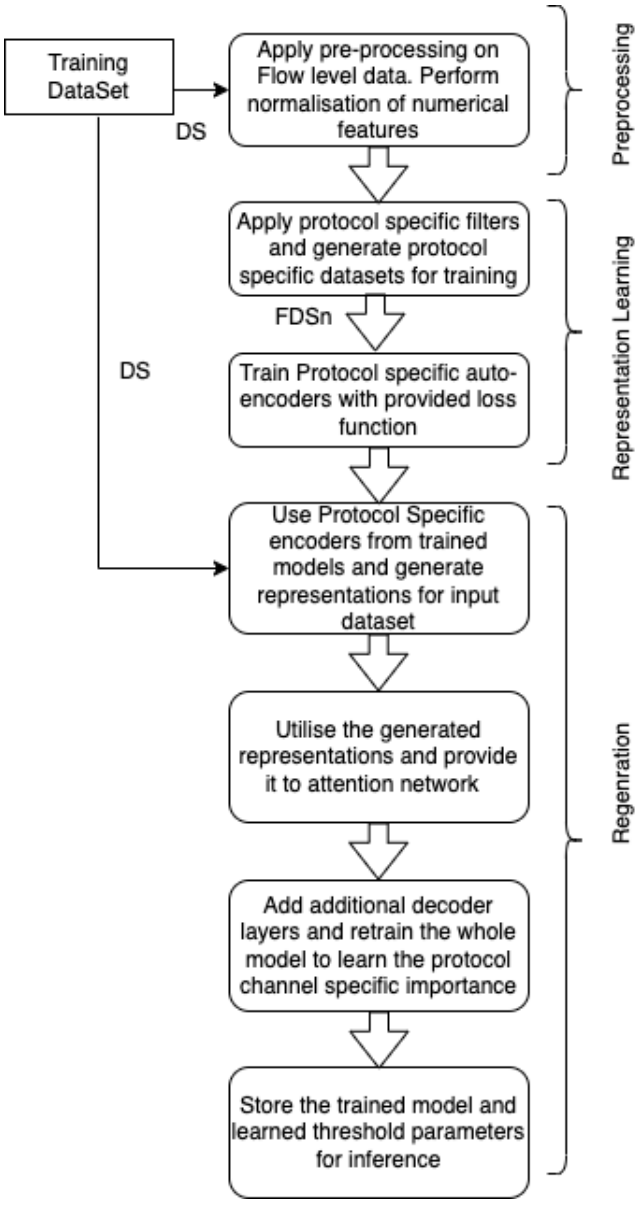

Figure 4.1: *Flow Diagram for the proposed protocol aware scheme*

representation is learned by training the autoenoder on global data (DS). Further, these individual protocol-specific representations are used with specific weight for a final decision in the regeneration stage and we call this decision as a protocol-aware decision. To learn the final model whole network requires to be retrained for incoming traffic (DS) using the regeneration error as a methodology for attack identification. To learn the weighted decisions of these individual protocol channels on the incoming traffic type, the encoder representations from each channel is utilized and processed with the attention network with an additional decoder layer to regenerate the input in the next stage. As described in figure 4.1, the proposed model has three conceptual stages - (i) Preprocessing, (ii) Representation Learning (global and local), and (iii) Regeneration, explained below.

4.3.1 Preprocessing

Table 4.1: *Flow level features used in the proposed protocol and non-protocol aware methods*

S.No.	Feature Category	Feature Description
1	Packet Transfer rate	Features related to forward and backward packet transfer rate
2	Packet Length	Features related to forward and backward packet length information
3	Inter-arrival Time	Elapsed time information between flows
4	Header information	Features specific to header information like flag count, header length etc.
5	Segment information	Features specific to segment size information
6	Transfer rate	Features specific to bulk transfer rate
7	Subflow related	Features specific to sublow information

This research work consider set of *network flow level records* which consist of bi-directional network flow-related statistical information such as *total packets, total size, packet rate, byte rate, inter-arrival time, TCP flag information, etc.* as available in dataset 4.1. These features are generally computed using flow analysis tools from the packet level information.

This research utilize the temporal context to derive the features with specific window length. To capture the temporal context the set of given flow records can be subsequently divided into fixed context window of size N based on timestamp and network flow level data can be assumed as a sequence of flow level records capture at different time instance. Let us assume input data as x which consists of sample records x_1, x_2, x_3, x_4, . . x_n generated at

subsequent timestamps t_1, t_2, t_3, t_4, . . t_n for the given time window of length N. This time window determines how much of the previous context needs to be captured for deciding the current state.

For each of these records associated with a particular time window, this research utilize the available numerical flow level features for each record and perform pre-processing for Data Preparation. Since the feature values follow different ranges, to avoid the model bias, it is required to convert them to a common scale. For this purpose, the Normalization is used to transform features in with mean and standard deviation as $(0, 1)$ respectively. If feature values are represented in x then the transformed value is $\hat{x}$, μ is mean and σ is the standard deviation of x and calculated on whole set of flow records, then equation 4.2 represents the normalization.

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{4.2}$$

From now on, a record x represents the transformed $\hat{x}$. With these given set of flow records, specific protocol specific filter is applied and data set is prepared for representation learning.

4.3.2 Representation Learning

Representation learning is the main component to learn the global and local representation of the data by applying various protocol-specific filters. Multiple representations are combined later to train the protocol-aware model. As shown in figure 4.2, in this stage each individual protocol-specific auto-encoder model (on protocol specific data FDS_i) is trained along with auto-encoder network without any filter (on global data DS). For Protocol-specific autoencoder specific data filter is applied and supplied to the autoencoder network for training. An Autoencoder consists of encoder and decoder module. The encoder module consists of multiple layers for processing the input data subsequently in multiple stages and each layer is associated with specific set of LSTM units to process. The flow of records is processed through each layer where the output of the next layer is based on the previous layer and dimension of data is reduced at subsequent stages. At the last stage of the encoder, an intermediate representation is generated which is provided to the Decoder module. The decoder module consists of the same number of stages as specified in the corresponding encoder layer and tries to reconstruct the supplied information at the subsequent stage of

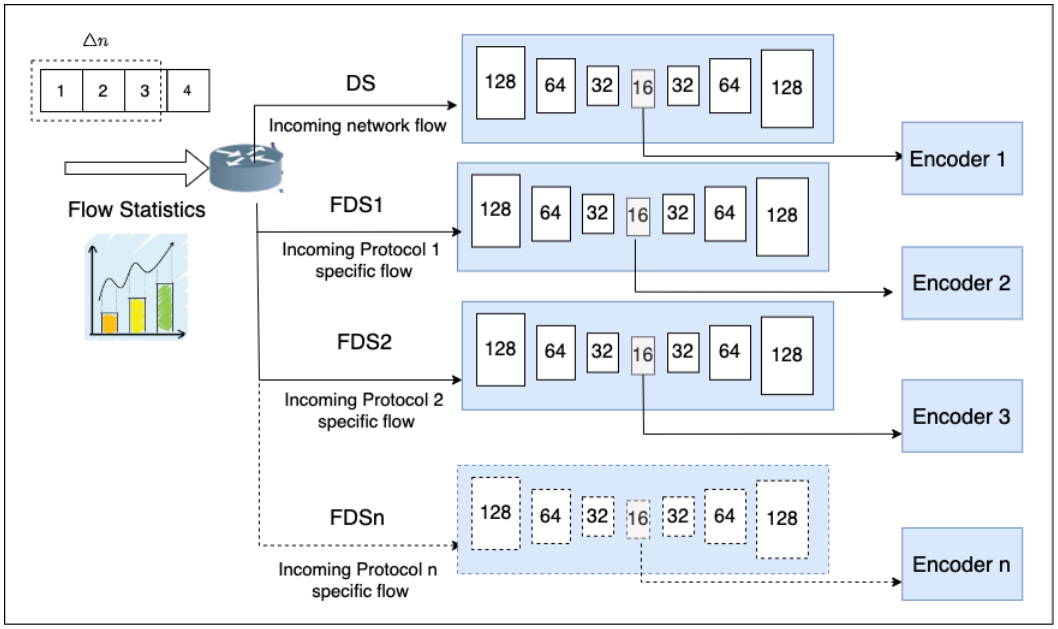

Figure 4.2: *Representation Learning. Each box in the encoder and decoder represent a processing layer of respective dimension.*

processing. At the final stage, Decoder generates the output as y which is of the same dimension as the input x. The final layer of the decoder generates an output vector and Mean Square Error (MSE) is used as a loss function to learn the intermediate representation for provided input. This loss between the input sequence x and output vector sequence y is represented using equation (4.3). For the provided context length of N the loss value MSE_x is estimated as :

$$MSE_x = \frac{1}{N}\sum_{n=1}^{N}|x_n - y_n|^2 \tag{4.3}$$

Once these protocol specific models are completely trained, encoder module is extracted and used for representation for incoming traffic data in next stage for regeneration.

4.3.3 Regeneration

The regeneration stage solves the main purpose of generating the same input as output and estimating the error values for normal traffic. Protocol-specific encoders trained in the previous stage are utilized for representation in this stage. In the normal scenario, data may contain very less records specific to the protocol, hence to handle the data sparsity during

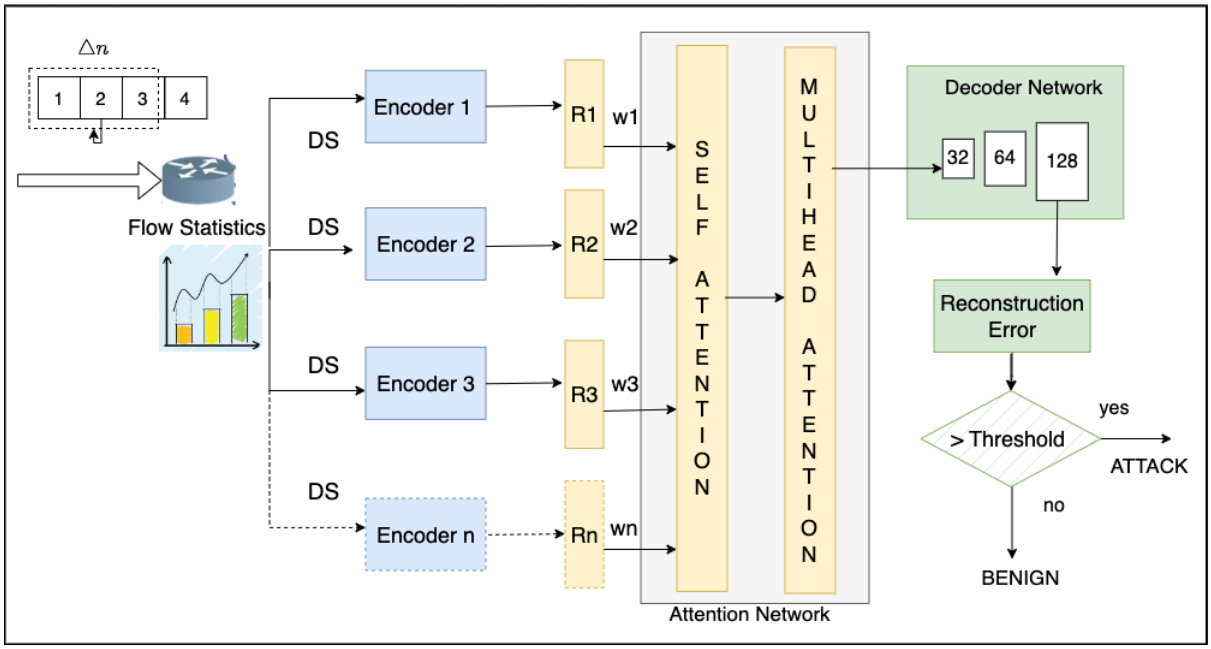

Figure 4.3: *Regeneration stage for the proposed protocol aware method*

inference, this research utilize the original data (without any filter) (DS) for training in this stage and compute the regeneration error. These error values ultimately help to identify the threshold θ which is used for attack detection.

In the regeneration stage, the proposed method utilize the encoder parts from protocol-specific auto-encoders trained in the representation learning stage and compute the representations (R_i) for supplied input as represented in figure 4.3. Now to learn the weights (w_i) of individual protocol channels for the supplied input, extracted representations are provided to the attention network consisting of the self-attention layer [125] and multi-head attention layers [95] in sequence to learn the importance of individual protocol channel and capture representation from different aspects.

It is expected that individual protocol-specific encoders will re-adjust the weights according to supplied input in this stage and provide more weight if input data belongs to specific protocol type and impose lower weights for others. In this stage learned encoders are used with attention and added decoder layers to provide improved results with the inclusion of channel level importance while deciding the kind of attack activity. The sequential input used in the representation learning stage produces the sequential representations which are assumed to be generated at subsequent timestamps. The Self-Attention mechanism for the

Table 4.2: *Information on Dataset parameters used in experimentation of Protocol-aware NIDS*

Dataset	Features	Class	Data length
CICIDS2018	74	Normal	667626
		FTP-BruteForce	193360
		SSH-BruteForce	187589
CICDDoS2019	64	Normal	30109
		DrDoS_NTP	1202642
		DrDoS_UDP	3134645
		SYN Flood	1582289

time sequential data, earlier specified in [95], is used to learn the effect of different positions of sequence in time in order to compute the final representation. The self-attention mechanism proposed earlier [125] for sequential data is utilized that consider the context for each timestamp.

As specified in the study [125], the attention weight between hidden states r_t and $r_{t'}$ (denoted by representations (R_i) in figure 4.3) at timestamps t and t' respectively for input x_t and $x_{t'}$ is defined as follows.

$$\alpha_{t,t'} = \sigma(w_a.\tanh(w_g.r_t + w'_g.r'_t + b_g) + b_a) \tag{4.4}$$

where σ is element-wise sigmoid function, w_g and $w_{g'}$ is weight matrices corresponding to their non linear combination. b_g and b_a are bias vectors. Further attention-focused hidden space representation l_t is computed (equation 4.5) for each respective channel for particular timestamp t where n is the context length. It quantifies how much to attend to particular channel at any timestamp on their neighbourhood context.

$$l_t = \sum_{t'=1}^{n} \alpha_{t,t'}.h_{t'} \tag{4.5}$$

At the next step Multi-head attention [95] layer is applied to capture the representation from multiple aspects that combines the knowledge of the same attention pooling via different representation sub-spaces and captures richer representations using multiple heads in parallel. Let the outcome from self-attention layer is output as l , original input vector is used as value x. Like in [95], the multi head attention is estimated in equation 4.6 where Query(Q), Key(K) and Value(V) vectors are represented as l, k, x .

$$\begin{cases} Attention(Q, K, V) = softmax(\frac{Q.K^T}{\sqrt{d_k}}).V \\ head_i = Attention(l.w_i^l, l.w_i^k, l.w_i^x) \\ M_{head(l)} = Concat(head_1, head_2, ...head_h).w^o \end{cases} \tag{4.6}$$

where d_k is the dimension of the l, and w_i^l, w_i^k, w_i^x, and w_i^o are learnable weight matrices. As shown in equation 4.6 first the representation of each $head_i$ is computed using input Query(l), Key(k) and Value(x) vectors and the weights w_i^l, w_i^k, w_i^x are learned respectively. The output from these multiple heads is concatenated together to compute multi-head representation $M_{head(l)}$. Now, $M_{head(l)}$ is passed as the input to the decoder layer to regenerate the input sample x. If $\acute{x}$ is the regenerated output vector sequence, the reconstruction error is defined as equation 4.7

$$Error_x = \frac{1}{N}\sum_{n=1}^{N} |x_n - \acute{x}_n|^2 \tag{4.7}$$

As mentioned above, the proposed model is trained with normal samples. While training the model for an arbitrary sample, the proposed method tests sample as attack or normal based on a threshold value θ, which is defined as the average of the regeneration error over M normal samples in the training dataset as in equation 4.8.

$$\theta = \frac{1}{M}\sum_{x=1}^{M} Error_x \tag{4.8}$$

Once the threshold is determined, incoming flow is identified to be an attack specifically if the reconstruction error is larger than threshold θ. It is expected that the reconstruction error for normal traffic data should be less as compared to that of anomalous traffic data.

An example of stage 2 processing is represented for set of Brute force attacks in figure 4.4. In this example, individual encoders trained on SSH, and FTP specific traffic is used in the first stage where each specific encoder captures the local representations for protocol specific benign traffic. And global representation is learned by training auto-encoder on non filtered benign traffic. As represented in figure 4.4 w1, w2, w3 are learned weights for individual channels (Non-Filtered, FTP,& SSH) using additional self-attention and multi-head attention layers. For the final model construction, whole network is fine tuned on

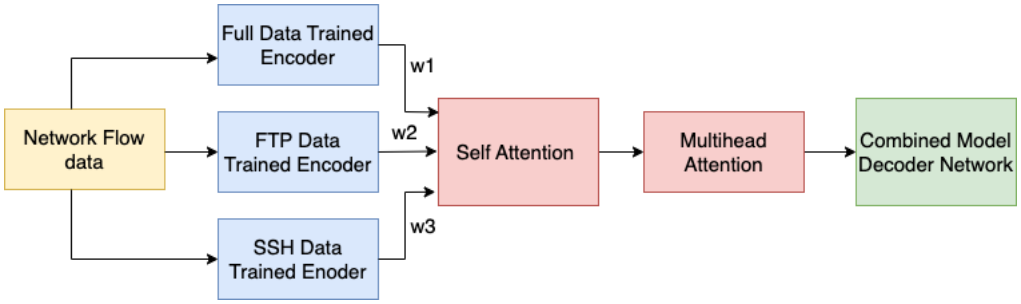

Figure 4.4: *Example of regeneration stage in the proposed protocol aware method.*

non-filtered benign traffic with additional decoder layers as represented in figure 4.3.

4.4 Experimentation Results

4.4.1 DataSet

To evaluate the performance of the proposed method, this research consider two different, publicly available, network attack-specific datasets i.e. CICIDS2018 dataset [1] and CICDDoS2019 dataset [2]. These are the recent dataset developed at Canadian Institute for Cybersecurity, where FTP-Patator and SSH-Patator tools are used for performing brute force attacks and LOIC tool is used for DDoS attacks. Network Flow records are generated on traffic data by using CICFlowMeter-V3 tool [102]. The characteristics of the datasets are shown in the Table 4.2. We utilized the flow level features as specified in table 4.1 for experimentation purpose where all the available features are used to train the model.

4.4.2 Implementation

This research utilize flow-level data for normal traffic captured for a specific dataset. All the available features for training as specified in table 4.1 are used for implementation. The complete implementation is performed on Macbook with 6 core intel core i7 processor with 16 GB RAM. The TensorFlow python package is used for the implementation and the self attention method as proposed in [125] is adopted for implementation. The multi-head attention implementation is used with 3 parallel heads. The configuration for an autoencoder network is used as shown in figures 4.2 and 4.3. The hyper-parameters used for autoencoder model is specified in table 4.3. The model performance is evaluated in terms of classification metrics by taking the timestamps specific to Attack or Benign class from

Table 4.3: *Autoencoder Model Configuration. The* **f** *indicates input feature vector.*

Parameter	**Values**
Total Layers	7
Encoder structure	**f**,128, 64, 32
Compressed output length	16
Decoder structure	32, 64, 128, **f**
Optimizer	Adam
Learning rate	0.0001
Loss function	Mean squared error
Batch size	32
Number of Epochs	50

the dataset description. The run time performance of model is also evaluated in terms of number of flows processed per second during inference time.

Since the proposed method requires the two-step process of training the protocol-specific autoencoders and utilizing the trained encoders in the regeneration stage to construct the protocol-aware model. To reduce the complexity, we can train the protocol-specific models in parallel and utilize them in the regeneration stage. The regeneration stage also allows configuring of the required number of heads in multihead attention to reduce the overall complexity.

The total complexity of the training an autoencoder model for two layers of size N is O(N^2) where N is the number of neurons in each layer. Let us assume there are k layers in autoencoder network then the overall complexity is O(k * N^2). Since k <<<N, therefore complexity can be assumed as O(N^2) . Since in the proposed methods the autoencoders are trained in parallel therefore for representation learning stage the complexity can be estimated as O(N^2). For Regeneration stage, the complexity of self attention and multihead attention (can be computed in parallel) is estimated as O(d_k * N^2) where d_k is the dimension of Query(Q), Key(K) and Value(V) vectors. Hence the overall complexity can be estimated as O(N^2) + O(d_k * N^2) .

4.4.3 Results and Analysis

4.4.3.1 Performance Comparison

This research work evaluates the detection performance of the models in terms of precision, recall, F1-score, and accuracy on different datasets as shown in table 4.4. To estimate the

Table 4.4: *Performance Comparison of Non Protocol Aware and Protocol Aware models.*

Dataset	Filter Type	Attacks Covered	Best Context Length	Prec.%	Rec.%	F1%	Acc.%
CICIDS2018	None	BruteForce attacks	2	92.7	100	96.2	93.8
	Proposed 1	BruteForce attacks	2	93.2	100	96.5	94.2
	Proposed 2	BruteForce attacks	2	**93.5**	**100**	**96.6**	**94.5**
CICDDos2019	None	DDoS attacks	10	94.7	**100**	97.3	95.1
	Proposed 1	DDoS attacks	4	95.1	98.5	96.8	94.3
	Proposed 2	DDoS attacks	4	**95.2**	99.7	**97.4**	**95.4**

Table 4.5: *Performance analysis of Protocol specific models.*

Dataset	Filter Type	Attacks Covered	Best Context Length	Prec.%	Rec.%	F1%	Acc.%
CICIDS2018	FTP	FTP-BruteForce	2	93.7	100	96.8	93.8
	SSH	SSH-BruteForce	10	93.3	100	96.5	93.9
CICDDos2019	UDP Port 123	DrDos_NTP	2	95.5	93.5	94.5	90.4
	UDP only	DrDos_UDP	2	95.7	100	97.8	96.1
	TCP/SYN	SYN Flood	4	95.2	100	97.5	95.6

prediction performance, 10% of normal data is hold out as testing along with the attack dataset, and estimate the evaluation metrics between *Benign* or *Attack* classes. The model experimentation is performed with different context lengths (up to 10) and report the best context length where the f1-score and accuracy are highest since different context length reports different performance figures in Table 4.4. This research estimates the performance for non-protocol specific, and proposed model (Without and with attention layer inclusion defined as Proposed 1 and Proposed 2 respectively) in table 4.4 .

From the results, it is evident that the proposed model (Proposed 1 and Proposed 2) outperforms the model created for global data (with Filter type 'None') both for CICIDS2018 and CICDDoS2019 dataset. It is also analyzed that performance for the protocol aware model is higher than non-protocol aware (Filter type 'None' in table) model as specified in table 4.4.

This research also estimated the performance by applying specific filters (With specific filters like FTP, SSH, TCP/SYN, etc.) and training protocol-specific models on the protocol-specific dataset. It is concluded that the specific model trained on filtered data alone performs better than the proposed method because of data and complexity reduction. The performance results for protocol-specific models are mentioned in table 4.5.

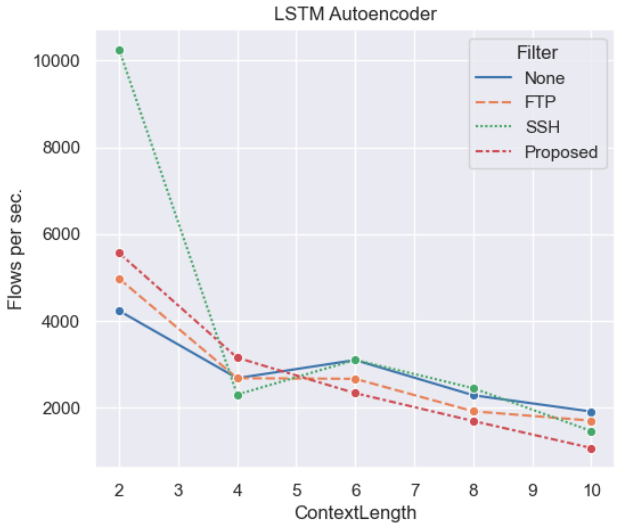

Figure 4.5: *Processing rate Vs Context length for CICIDS2018 dataset*

4.4.3.2 Context Length vs Processing rate

Further, this section also investigates the processing rate in terms of flow records per second processed by the trained model against the context length for protocol specific models and proposed method (with attention network). This research work estimated the results on specific datasets and the set of attacks as discussed earlier as shown in figure 4.5 and 4.6. The figure indicates that the flow processing rate is reduced by taking a larger context length which happens due to higher processing overhead. This analysis can help us to specify an optimum context length for processing the records in an estimated time.

4.5 Summary

This research work proposed the method for building an intrusion detection solution that is protocol-aware and unsupervised in nature as it requires only Normal traffic records for training the model. The proposed system also reduces the overhead of labeling every network flow data for a specific attack. The implementation of intrusion detection is based on the latest & representative flow-level dataset available in the open-source community. Various results by utilizing different processing units in an inherent autoencoder-based network are presented and further estimated the processed flow per second by these individual units. From the results, it is evident that the proposed model outperforms the non-protocol-aware

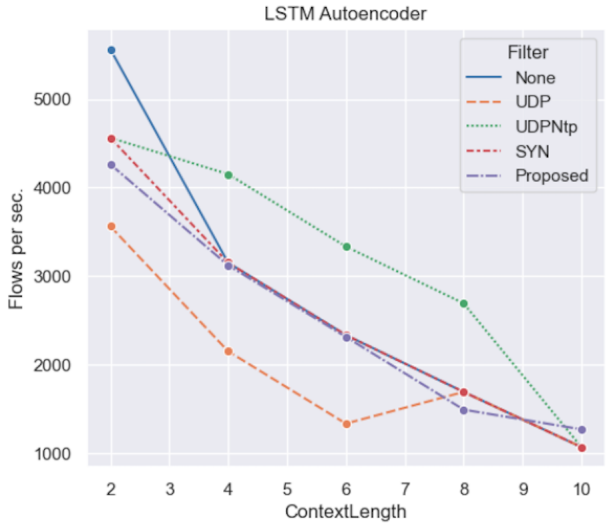

Figure 4.6: *Processing rate Vs Context length for CICDDoS2019 dataset*

model. In the next chapter, network packet level data is utilized to generate multiple views that are used to build an unsupervised learning based IDS solution.

5

Mutliview-based Network Intrusion Detection System

5.1 Introduction

The previous chapter discusses the approach for building protocol protocol-aware unsupervised intrusion detection systems by using the autoencoder-based model. In the earlier proposed solution, the flow level information present in existing datasets is considered. However, IDS can also be built on multiple other representations generated from network packets. Most of the machine learning-based intrusion detection systems consider training a model from the extracted network packet information. However, these network packets can also be transformed into multiple different views like network headers, flow level information, binary files like images, text, etc. In this chapter, a multi-view-based intrusion detection method is proposed that is based on self-supervision to detect the attacks from raw network packets by training the separate auto-encoder models for each view using only normal traffic. Subsequently, utilizing reconstruction error to build the self-supervised model based on the majority voting of individual view-based autoencoder's outcome.

5.2 Need for Multi-view based NIDS

In the recent era, various available open-source utilities like LOIC [5], Slowloris [6], Nmap [7], etc. are used by attackers to perform multiple different types of network attacks easily that can hamper the overall communication between the entities and affect the system performance. This communication between the hosts depends on the type of protocol used at the application level. For most of the networking applications, like FTP (File Transport Protocol), SSH (Secure Socket Shell), etc. which are based on connection-oriented protocol i.e. TCP (Transmission Control Protocol), the connection is first established and data transfer takes place later, due to which flow level data is significantly generated and communication is well represented in the forms of flows. On the other hand for the applications like NTP (Network Time Protocol), SNMP (Simple Network Management Protocol), etc. which are based on connection-less protocols i.e. UDP (User Datagram Protocol), the prior connection establishment process is not required due to which the flow level details are minimally generated and packet headers are able to describe the connection information well. Hence, in the Intrusion Detection System (IDS) perspective, the individual view depicts its own importance depending upon the kind of protocol used for a particular type of application.

On the other hand, today most of the traffic is encrypted and most of the IDS solutions are not suitable for deep packet inspection. Therefore, to effectively develop an IDS solution for modern encrypted network traffic, payload information is not sufficient for detecting attacks and it is required to utilize information from other sources like header and flow statistics [126] for accurate detection. Several IDS solutions like [67, 127, 128] utilizing the flow level information have been developed earlier to resolve this problem. From another perspective, information used for building IDS solutions also plays a vital role in determining the kind of attack that can be detected by IDS easily like, most of the rule-based IDS solutions (i.e. Snort [21], and Bro [22]) consider header fields and payload distributions details only for attack detection, due to which payload-related attacks are not captured. In other solutions, header and payload are utilized together but they are unable to detect low rate attacks. Other proposed solutions like [78, 129, 130] consider only the payload information and do not utilize the information from header or flow level data due to which the information from flow sequence is missing and such solutions are unable to detect the specific attacks. Due to the aforementioned reason, the model built on individual views can capture different characteristics, and information from multiple representations can

be combined for the accurate detection of attacks. With the combined knowledge, the advantage of each view can be utilized for attack determination. Due to these reasons, a system is required where all the aspects are needed to be looked into for accurate detection. This research work proposes a method of utilizing the views from different perspectives.

Because of the aforementioned reason, the model built on individual views captures different characteristics. And predictions from individual views might not overlap with one another. This property can help us to build a combined model with improved overall performance and can help us to learn differentiated features by considering the multiple views together.

5.2.1 Idea & Contributions

This research consider a self-supervised framework that uses the regeneration error from multiple view-based autoencoder models for automatically labeling the incoming traffic. The proposed method utilizes the network packet information and multiple views from packet-level data are first constructed on which individual view-based representations are learned through unsupervised reconstruction error based mechanism. Now to aggregate the information from multiple views together, intermediate representations from the trained models are utilized to classify the input data (i.e. Benign or Attack) based on the view-specific learned reconstruction error thresholds. These regeneration error thresholds are dependent on network traffic properties due to which a minimal change in thresholds is also required to be handled. In order to solve this problem the proposed method has used the self-supervised learning-based approach by taking the majority voting from individual view-based models and generating the labelled data with small change in thresholds. The proposed approach further uses this labeled data in the final stage to train the binary classification model to predict the outcome as Benign or Attack. From various experiments over the CICIDS2018 [1] dataset, it is evident that the proposed self-supervised learning-based multi-view model outperforms its baseline counterparts for attacks like the FTP BruteForce attack & UDP DDoS attacks. The major contributions of this research can be summarized as follows:

- This research work presented a mechanism to generate multiple views from raw packet-level data and proposed multiple view-specific features that are utilized for training view-specific autoencoder models. The proposed multiple views are captured at a

given fixed window of time.

- The proposed system is based on a self-supervised learning approach that constructs labels for the incoming data with a majority voting of trained individual view-specific encoders.
- The labeled data is ultimately utilized to build a binary classification model, that can be used to detect attacks in incoming traffic.
- A recent benchmark dataset i.e. CICIDS2018 [1] is used for evaluating the performance of the proposed model on FTP Bruteforce and UDP DDoS attacks and compare results for view-specific auto-encoders individually with the proposed method.

The Chapter is organized into different sections as follows. In Section 5.3, the various solutions proposed on different views derived from network packets for attack detection are discussed. Section 5.4 describes the proposed methodology and algorithm in detail and the components involved. Section 5.5 presents the implementation of the proposed approach, dataset, & experimentation results. In section 5.6 the summary is provided.

5.3 Related Work

As discussed earlier in section 2.1, Network packets consist of two important parts i.e. Network header (describing the protocol header-specific information) and Network payload (consisting of the data bytes transferred). These network packets can be further aggregated based on various parameters, and statistical features like byte per sec, inter-arrival time, sub-flow information, etc can be computed to build the flow level information. Other representations can also be generated similarly with the further transformation of packets to binary formats like Images [131], Text [132], etc. This research work study various IDS solutions proposed earlier that consider multiple views like network header 5.3.1, flow 5.3.2, image 5.3.3, generated from network packets. A brief description of different methods is presented as follows:

5.3.1 Network view based solutions

Network view-based solutions consider the information extracted from network headers. Most of the earlier proposed solutions PHAD [133] and SPADE [134] consider building profiles for network header fields and detecting attack activity by comparing them with

the pre-determined thresholds. Another proposed method [135] considers the statistical analysis of packet header data to detect attacks. It is based on analyzing the correlation of destination IP in outgoing traffic and uses wavelet transforms to study the address and port number correlation for different timescales. Most of these proposed solutions are based on outdated datasets and are not relevant to recent attacks.

5.3.2 Flow view based solutions

Most of the Flow-based solutions proposed earlier consider a supervised learning-based approach like Tomio [136] proposed a multiple flow features-based classification model that considers different views of MAWIFlow dataset [137] like Nigel, Orunada & Viegas features. It consists of two main steps of classification with an autonomous model update, where the classification decision is based on the consensus of random forest models trained for each set of features using majority voting. Various deep learning-based solutions are also proposed to mitigate the problems of supervised learning-based models. For instance, autoencoder-based solutions proposed by [27,28,113,114] and similarity learning-based methods proposed by [117,118] are discussed earlier that utilize flow level information, but these models suffer from high computation time and complexity.

5.3.3 Image view based solutions

Most of the other proposed methods also consider representing network packets in binary format (primarily image) and developing the classification model on the represented information. Irina [138] proposed a malware detection method based on binary visualization and self-organizing incremental neural network (SOINN [139]). The algorithm chosen for binary representation uses Hilbert space-filling curves [131] for a better imprint of the image. The proposed method first converts the binary files into an image and then considers 4 parts of an image as the region of interest to construct the histogram with a feature-length of 1024 and utilized the SOINN model for malware detection. The dataset used for experimentation consists of 2K benign files from a trusted source and 2K malicious files collected from the VirusShare website. The detection accuracy of 91.7% and 94.1% is achieved for ransomware attacks in .pdf and .doc files. Another method for IoT malware traffic detection is proposed by Robert et. al [140], which considers transforming network traffic into 2D images using the binvis.io [141] utility and classifying the images using the MobileNet model. In the

first step network traffic is captured using a sniffer and then to convert the binary file to 2D image data is assumed as a byte string and converted to specific color using an ASCII table. The proposed approach achieved an accuracy of 91.32%. Another similar method proposed by Joseph [121] for IoT network attack detection uses a combination of network traffic profiling and machine learning components. The Network profiling component uses rate metrics for calculating the deviation from normal profile due to attack and the machine learning component built on MobileNetV3 [122] classification model uses extracted images from packet capture (pcap) files for attack detection. The combination of both systems results in an average accuracy rate per iteration of 99.17%. Most of these proposed solutions are based on supervised learning mechanisms that require labeled data for training.

5.3.4 Multi aspect based solutions

Various approaches are also studied that consider multiple aspects generated from the same feature set and build supervised learning-based models for detection. For example, the classification method proposed by Ming [49] considers spatial and temporal views together. It consists of 3 layers parallel network structure where the first and third sub-networks consider spatial features using a convolution neural network model (CNN) based model and the second network simultaneously learns spatial and temporal features using the CNN-LSTM network. The proposed network learns the network using early fusion technology to improve the accuracy of network traffic classification. This proposed method is tested on ISCX2012 and CICIDS2017 datasets and provides a classification accuracy of 99.9%. Another solution HCRNNIDS [50] is proposed earlier to initially process the features through CNN and later passed through recurrent neural network (RNN) layers to generate the sequence at each time step. CNN is used to capture the local features while temporal features are captured using RNN. This method is applied to the CICIDS2018 dataset and detects attacks with an accuracy of 97.75%.

The major limitation of the earlier proposed methods is that it considers a single view or set of features at a time and multiple aspects from one type of features are utilized only in the supervised learning-based solution. To mitigate these issues, our proposed model considers multiple views together to build an IDS solution that is based on a self-supervised learning mechanism and avoids the requirement of labeled data for model training. In the

next section, the proposed methodology is discussed in detail.

5.4 Proposed Methodology

The proposed method considers the raw packets from network communication. Various packet sniffing tools like Wireshark, tcpdump, [142] etc. can be used to capture these raw packets from the port in dedicated machine or network switch that is running in promiscuous mode. These raw packets contain communication information among different hosts. For the experiment purpose, this research used the raw packets already extracted for specific attacks, but in the real network, a sniffing module is required to capture the packets consecutively with a pre-defined time window length.

As shown in the flow diagram 5.1, the extracted network packets are first converted to multiple different views using the subset of packets captured at fixed time window length and later the view-specific autoencoder models are trained and view-specific reconstruction error thresholds are estimated by training the autoencoder models with reconstruction error as a loss function. These trained encoders (part of autoencoder) are further utilized for generating representation from the extracted views and used for producing labeled data with the majority voting on view-specific encoder classification results. This label generation process also handle the scenario for the slight change on the threshold values to build the more robust model that can handle the minimal change in learned threshold as the value of threshold may vary over time due to change in network characteristics. In the final stage of processing, the prepared labeled data is utilized to build the binary classification model that is used to predict the outcome as Benign or Attack.

The proposed solution for the intrusion detection system consists of various modules of processing as shown in the High-level architecture diagram 5.2 for the proposed model training method. Section 5.4.1 presents how various views are generated from network packets, Section 5.4.2 discusses the details on training view specific auto-encoders and section 5.4.3 presents the self supervised model training process. Section 5.4.4 discusses the algorithms for training and inference process in detail.

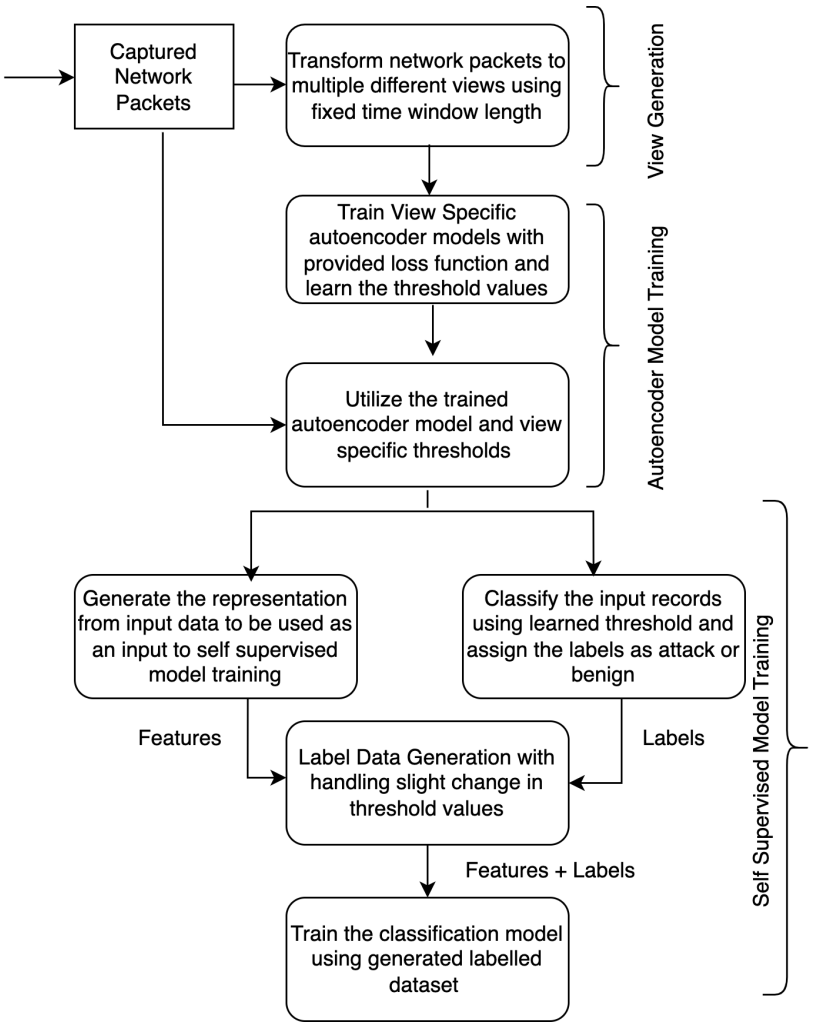

Figure 5.1: *Flow diagram for Multi-View based IDS model training*

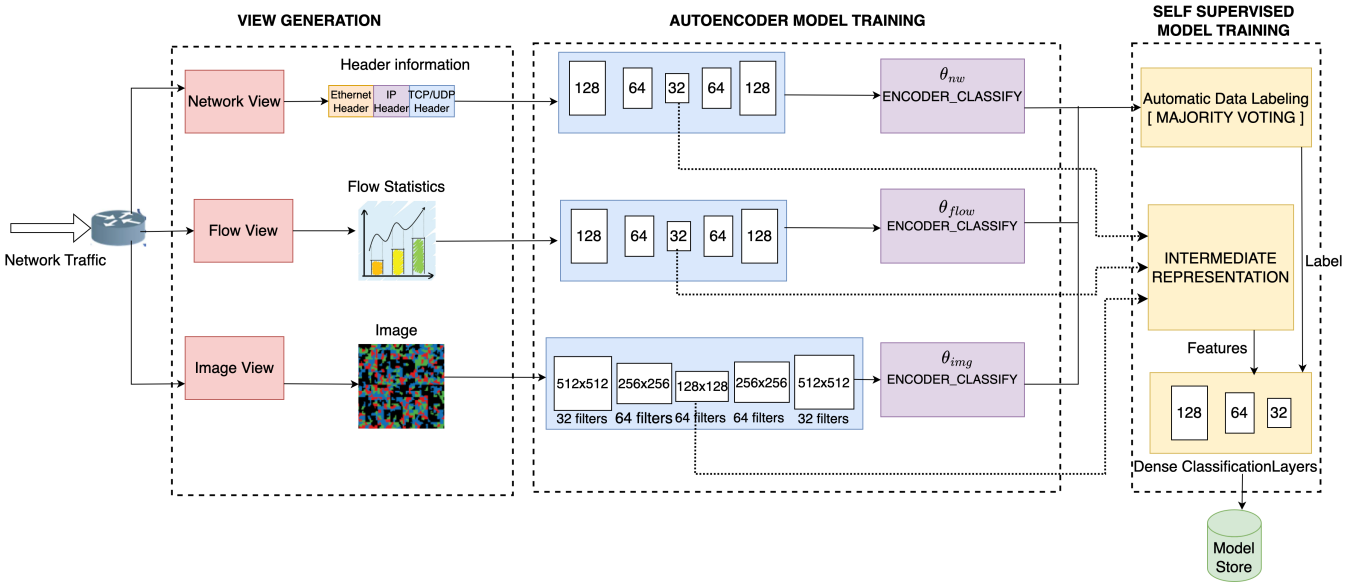

Figure 5.2: *High Level Architecture for Model Training*

5.4.1 View Generation

Once the packets are captured from the packet sniffing module, various views are extracted in View generation module. For a pre-defined time window of $\triangle t$, the proposed method first extracts the batch of network packets based on generated timestamp and uses these batches to construct multiple different views as follows:

5.4.1.1 Network View

Network view is constructed from the network header part of the networking packets. The associated network header consists of information from various protocols from TCP/IP networking layers. These header details are used to construct network view that consists of information about protocol headers, aggregated volume statistics, etc. Various protocol related features are extracted from this information and aggregated values are computed with pre-defined time window $\triangle t$ as specified in the table 5.1

5.4.1.2 Flow View

Flow view is constructed by using the flow-level information derived from the communication among the entities in the network. It consists of statistical features computed on overall

Table 5.1: *Feature Information for Network View*

S.No.	Feature Category	Feature Description	Count
1	Interface Based	Features related to transferred octets incoming and outgoing, incoming and outgoing unicast and non-unicast packets i.e. interfaceInOctets, interfaceOutOctets, interfaceInNUcastPkts, interfaceOutNUcastPkts, interfaceInUcastPkts, interfaceOutUcastPkts	6
2	IP header based	Features related to incoming and outgoing IP datagrams i.e. ipInReceives, ipOutRequests	2
3	ICMP header based	Fearures related to number of incoming and outgoing icmp total, echo, destination unreachable messages i.e. icmpInMsgs, icmpOutMsgs, icmpInDestUnreachs, icmpInEchos, icmpOutDestUnreachs, icmpOutEchoReps	6
4	TCP header based	Features specific to TCP header information like incoming and outgoing segments, connection information i.e. tcpPassiveOpens, tcpOutSegs, tcpOutRsts, tcpCurrEstab, tcpActiveOpens, tcpEstabResets.	7
5	UDP header based	Features constructed from UDP header information like input and outgoing datagrams i.e. udpInDatagrams, udpOutDatagrams, udpNoPorts	3

transfer characteristics specific to established connections between hosts. Flow normally denotes 5 tuple information i.e. $< srcIp, dstIP, srcPort, dstPort, protocol >$. These features can be easily captured by using various utilities like Netflow [106], CICFlowMeter-V3 [102] etc. CICFlowmeter-V3 utility is used in this work to capture the flow level information. The generated flow view consists of various statistical features like packet transfer rate, inter-arrival time, flow activeness etc. as specified in the table 5.2

5.4.1.3 Image View

Image view is constructed by using raw network packets which contain both network header and payload information together. The proposed method constructs the image view from captured raw packets at fixed time interval $\triangle t$ by using binvis.io [141] utility. This utility can also help us to look for suspicious parts in packed or encrypted files like binaries, and to locate relevant pixel offsets. It provides a visual overview for easier orientation and deeper insights. This utility internally uses Hilbert Space filling curve [131] for clustering the data which generates a better imprint of an image. In this algorithm, the image data is divided into multiple bytes and each byte is assigned to a specific color code depending on its ASCII

Table 5.2: *Feature information for Flow view*

S.No.	Feature Category	Feature Description	Count
1	Packet Transfer rate	Features related to forward and backward packet transfer rate	5
2	Packet Length	Features related to forward and backward packet length information	15
3	Inter-arrival Time	Elapsed time information between flows	14
4	Header information	Features specific to header information like flag count, header length etc.	14
5	Segment information	Features specific to segment size information	4
6	Transfer rate	Features specific to bulk transfer rate	7
7	Subflow related	Features specific to sublow information	7
8	Flow activeness	Statistical Features specific to flow activeness information	8

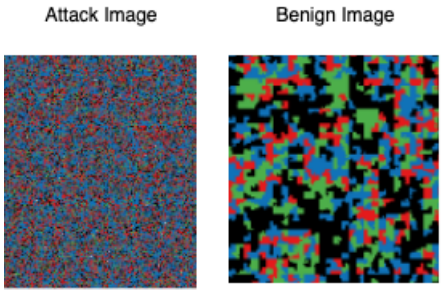

Figure 5.3: *Image View for Benign and Attack Scenario*

values as follows:

- blue for printable character
- green for control characters
- red for extended characters
- black for the null character
- white for the non-breaking space

It is also observed that images related to attack consist of dense pixel values with red and blue regions, which is easily distinguishable from normal scenarios where most of the images consist of sparse black regions as shown in figure 5.3. It also represents that amount of packets generated during the attack is more due to which dense regions are created.

Once the views are generated, the information is represented into numerical features. But, these numerical features can follow a different range of values. So to avoid model bias, it is required to convert them into a common scale. For this, Normalization is used to transform features in with mean and standard deviation as $(0, 1)$ respectively. Let us assume the feature values for a specific view is represented in vector $\hat{X}^i$ then the transformed value is X^i is the normalized representation computed through equation 5.1 where, μ is mean and σ is the standard deviation of $\hat{X}^i$. For the Image view, each pixel value is divided by 255 so that values are aligned in the range from [0,1].

$$X^i = \frac{\hat{X}^i - \mu}{\sigma} \tag{5.1}$$

5.4.2 Auto-encoder Model Training

Autoencoder Model Training module first performs the job of training individual view-specific auto-encoder models for normalized view specific features (X^i), and learn the reconstruction error thresholds specific to each view. To build the view-specific auto-encoders on provided data, this work assumes input data for view is X^i which is represented as a set of N records x_1, x_2, x_3, . . . x_N. Each of these input records (represented with normalized features) is supplied to the Encoder network. The encoder module consists of multiple layers and each layer is associated with multiple processing units. These units are selected based on the type of each view. For instance, Multilayer Perceptron (MLP) units are used for network and flow view (1 dimensional) and CNN units are used for Image view (2 dimensional). The flow of records is processed through each encoder unit. At the last stage of the encoder, an intermediate representation is generated which is provided to the Decoder module. The decoder module consists of the same number of stages as specified in the corresponding encoder layer and tries to reconstruct the supplied information at the subsequent stage of processing. At the final stage, Decoder generates the view-specific output as Y^i which is of the same dimension as the input X^i. The final layer of the decoder generates an output vector and Mean Square Error (MSE) is used to calculate the error between the input and output vector using equation (5.2). This error helps us to quantify the view specific reconstruction error threshold θ^i for normal traffic which is based on the mean of the error values as shown in equation (5.3). Once the threshold is determined, incoming flow is identified to be an attack specifically if the reconstruction error is larger

than threshold θ^i. Let D is the feature dimension for view X^i and N is the total number of training records then the error value $Error^i$ and threshold θ^i is estimated as

$$Error^i = \frac{1}{D}\sum_{d=1}^{D}|X^i(d) - Y^i(d)|^2 \tag{5.2}$$

$$\theta^i = \frac{1}{N}\sum_{n=1}^{N} Error^i(n) \tag{5.3}$$

After this stage, the set of various thresholds (θ^1, θ^2, ... θ^n) are learned for each specific view, which is used to classify the input records. These individual thresholds are used to label the individual records Benign or Attack in the next stage.

5.4.3 Self-supervised model training

Automatic data labelling is the next step that delivers self-supervision for the proposed scheme. It is used to generate the attack or benign labels from the pre-trained view-specific autoencoder models. This stage utilizes the view-specific thresholds learned in the previous stage to compute the label for an outcome of each record when passed through a learned autoencoder. For a given record, multiple views are processed through each view-specific encoder separately to identify the attack activity. Now the final label specific to the network event is assigned as the majority vote of the individual encoders. As shown in figure 5.4 majority of outcome from individual view-based autoencoder is assigned as a final label (anomaly as True or False) for further training. In this example, loss values and thresholds for network, flow and image view are defined as (loss_nw, loss_flow, loss_img) and (threshold_nw, threshold_flow, threshold_img) respectively. This stage also handles the slight change ($\theta^i \pm \delta$) in the values of learned view specific thresholds by estimating the slightly modified threshold and regenerating the labels. By considering the slight change in threshold, the trained model becomes more robust towards the minimal changes in network characteristics.

For the majority voting, Let $C_1(X)$, $C_2(X)$... $C_n(X)$ be the predicted classes from n different view-based encoder models for an input vector X then, final label $C_{major}(X)$ can

	loss_nw	threshold_nw	anomaly_nw	loss_flow	threshold_flow	anomaly_flow	loss_img	threshold_img	anomaly_img	anomaly
0	0.389049	0.220287	True	0.287950	0.221312	True	0.101505	0.096446	True	True
1	0.390333	0.220287	True	0.769167	0.221312	True	0.101798	0.096446	True	True
2	0.315964	0.220287	True	0.054242	0.221312	False	0.086322	0.096446	False	False
3	0.047842	0.220287	False	0.052423	0.221312	False	0.089829	0.096446	False	False

Figure 5.4: *Sample of automatic data labeling using majority voting*

Table 5.3: *Model Configuration for Binary Classifier.*

Parameter	**Values**
Additional dense Layers	3
Concatenated Encoder Representation Length	96
Layer Structure	128, 64, 32
Optimizer	Adam
Learning rate	1e-4
Loss function	Binary cross entropy
Batch size	16
Number of Epochs	50

be computed with equation 5.4

$$C_{major}(X) = mode(C_1(X), C_2(X), ...C_n(X)) \tag{5.4}$$

The data prepared using self-supervision is used now to build the binary classifier. In this stage, the proposed method utilize pre-trained view-specific encoders from learned autoencoders with added deep layers to train the classifier model. View-specific encoders are first used to encode the incoming view and generate representation. These representations are further concatenated together and supplied to classifier consisting of fully connected layers with softmax layer as an output. This binary classifier consists of 3 dense layers with 128,64,32 neurons and utilize binary cross entropy loss function for training the model. More details on the learning structure of the network can be found in table 5.3

5.4.4 Algorithm

The proposed algorithm involves multiple stages of processing as described earlier in figure 5.2. These stages play a vital role in the overall process. The Pseudo-code for the proposed model training and inference is represented in algorithm 1 and algorithm 2 respectively.

5.4.4.1 Model Training

Algorithm 1: Model Training

Input : Time window $\triangle$T, δ, and collection of raw packets P
Output: Classification Model M, and Autoencoder models $A^{network}$, A^{flow}, A^{image}

$X^{network}, X^{flow}, X^{image} \leftarrow VIEW\_GENERATION(P, \triangle T)$
$X \leftarrow X^{network}, X^{flow}, X^{image}$
for $X^i \in X$ **do**
  $A^i, \theta^i \leftarrow AUTOENCODER\_MODEL(X_i)$
end for
Initialize label array as L
for $P^j \in P$ **do**
  Initialize class array as C
  for $X^i \in X$ **do**
    $C^i_j \leftarrow ENCODER\_CLASSIFY(\theta^i \pm \delta, P^j)$
    Add C^i_j to C
  end for
  $L^j \leftarrow MAJORITY\_VOTING(C)$
end for
Model $M \leftarrow CLASSIFICATION\_MODEL(P, L)$

The algorithm 1 presents the training process in detail. For the pre-selected time window $\triangle$T and raw packets P, different views $X^{network}$,X^{flow} , X^{image} are generated at an initial stage. The view generation functionality is represented as $VIEW\_GENERATION()$ which is used to create multiple views as per sub-section 5.4.1. Later, for each of the views, separate autoencoder models are trained, and view-specific thresholds θ^i are determined. The autoencoder training process is defined as function $AUTOENCODER\_MODEL()$ that trains the model on supplied view X_i. Now with each autoencoder, an input record is predicted as Normal or Benign depending on comparing calculated loss with learned view-specific threshold. The process of prediction is called $ENCODER\_CLASSIFY()$ that takes the learned thresholds and the variation factor δ as input. Additional records are generated by including the slight change (δ) in threshold values. Further, the majority voting is applied on predicted class metrics, and the final label is identified for an input record. The associated function $MAJORITY\_VOTING()$ considers the outcome from each view-specific autoencoder and selects the majority as a final outcome. This information is generated for all input records. In the final stage, the binary classifier M is trained

with labeled data and stored for the inference process. The classification model is trained in function $CLASSIFICATION\_MODEL()$ by using the final labels generated at the previous step and using the specified configuration in table 5.3. The whole training process is explained below in the algorithm 1.

The overall complexity of the training process depends upon the associated functions. The $VIEW\_GENERATION()$ function considers calculating the aggregated features for the set of packets in the specified time window. The specified views can be computed in parallel with linear complexity therefore for V such views and F features, the total complexity is O($V * F$). The complexity of training an autoencoder model with a constant number of layers and with N computation units is around O(N^2) and multiple view-specific autoencoders can be trained in parallel too. Function $ENCODER\_CLASSIFY()$ also takes O(N^2) time for encoding the view-specific information. The final classifier trained is a multi-layer perceptron and its complexity is also O(n^2) for n computing units. Hence the overall complexity for the model training process for a particular batch is around O($V * F$) + O(N^2) + O(n^2).

5.4.4.2 Model Inference

The algorithm 2 represents the execution of the proposed scheme during inference where the trained models are first loaded from persistent storage. It utilizes the view-specific autoencoder models for encoding purpose and uses the classification model directly to classify the generated view-specific representations as Benign or Attack. In the first step, the packets captured at a pre-defined time window $\triangle T$ are used to generate the multiple views iteratively by using the function $VIEW\_GENERATION()$. Later these views are processed from the respective encoders to generate the representations (i.e.$R^{network}$,R^{flow},R^{image}). This encoding is represented as a function $ENCODE()$ that takes a trained autoencoder and view as an input and generates the view-specific representation. These representations are further classified from the trained self-supervised model to predict a Benign or Attack scenario and are represented as a function $CLASSIFY()$.

The overall complexity of the proposed model inference algorithm depends on the functions $VIEW\_GENERATION()$, $ENCODE()$, and $CLASSIFY()$ functions. For an incoming set of packets captured at time window $\triangle$T, V views are generated for F features so the complexity is around O($V * F$). The function $ENCODE()$ is used to encode the view information through autoencoder which is O(N^2) for N computation units. and

$CLASSIFY()$ function generally takes O(n^2) for n computing units. Therefore the overall complexity of algorithm is O($V * F$) + O(N^2) + O(n^2).

Algorithm 2: Model Inference

Input : Model M, Time window $\triangle$T, $A^{network}$,A^{flow}, A^{image} and collection of raw packets P

Output: Benign Or Attack

1: Load models M, $A^{network}$, A^{flow}, A^{image}
2: **for** $P^i \in P$ **do**
3: $X^{network}, X^{flow}, X^{image} \leftarrow VIEW\_GENERATION(P^i, \triangle T)$
4: $R^{network} \leftarrow ENCODE(A^{network}, X^{network})$
5: $R^{flow} \leftarrow ENCODE(A^{flow}, X^{flow})$
6: $R^{image} \leftarrow ENCODE(A^{image}, X^{image})$
7: Benign or Attack ←CLASSIFY(M, R^{network}, R^{flow}, R^{image})
8: **end for**

5.5 Experimentation Results

5.5.1 DataSet

To evaluate the performance of the proposed method, the CICIDS2018 dataset [1] is considered which is a publicly available dataset developed at the Canadian Institute for Cybersecurity (CIC) where FTP-Patator and SSH-Patator tools [100] are used for performing brute force attacks and the LOIC, HOIC tools are used for performing DDoS attacks. To prepare the dataset, network flow records are generated on traffic data by using CICFlowMeter-V3 tool [102] and in total 7 different types of attacks are captured at different dates. It also contains data in raw network packet files also which is only used for experimentation purpose. The raw packets for which the FTP BruteForce attack and UDP DDoS attack are performed along with Benign traffic are considered in the experimentation. The raw packet data specifically for the victim host (IP address: 172.31.69.25) is used on which various views are constructed to build the model. The selected pcap files consist of 4718327 number of packets in total for FTP brute force-specific attacks performed on Wed-14-02-2018 date and a total of 91666016 packets for UDP DDoS-specific attacks that was preformed on Tue-20-02-2018 date.

Table 5.4: *Model Configuration for Network and Flow View. The* **f** *indicates input feature vector.*

Parameter	**Values**
Total Layers	9
Encoder structure	**f**,128, 64, 32
Compressed output length	16
Decoder structure	32, 64, 128, **f**
Optimizer	Adam
Learning rate	0.0001
Loss function	Mean squared error
Batch size	16
Number of Epochs	50

5.5.2 Implementation

For the implementation purpose, the proposed method extracted data for the Benign scenario specific raw packets by utilizing the timestamp information, when the attack has not been performed and generated various smaller .pcap files using editcap utility [143] with the defined time window length. In various experiments, we increase the time window size (from 1 min. upto 4 min.) and analyze how the model performs with the increment as the time window affects the amount of data generated after aggregation. These binary files (.pcap) are used further to generate multiple views, where binvis.io [141] is used for image view generation, CICFlowMeter-v3 [102] is used for preparing flow view and own implemented code is used for generating network view. Further, view-specific autoencoder models are trained and view-specific thresholds are identified for each view individually. The complete implementation is performed on Mac OS with 6 core intel core i7 processor with 16 GB RAM and utilized the TensorFlow python package for the implementation. The configuration for autoencoder networks for different views are used as shown in Table 5.4 & 5.5. The model performance is evaluated in terms of classification metrics by considering the timestamps specific to Attack or Benign class from the dataset description. The run time performance is also evaluated for the proposed model in terms of the number of flows processed per second in run time. The $\delta = 0.10$ is used as a value to consider the change in threshold parameters while training the self supervised model.

Table 5.5: *Model Configuration for Image View.*

Parameter	Values
Total Layers	6
Input Dimension	512x512x3
Encoder structure	512x512x32, 256x256x64
Compressed dimension	128x128x64
Decoder structure	256x256x64, 512x512x32
Optimizer	Adam
Learning rate	0.0001
Loss function	Mean squared error
Batch size	16
Number of Epochs	50

5.5.3 Results and Analysis

5.5.3.1 Performance Comparison

The detection performance of the proposed model is evaluated in terms of precision (Prec.), recall (Rec.), F1-score (F1), and accuracy (Acc.) for various time windows (1 min, 2 min, 3 min, and 4 min). To estimate the prediction performance, 10% of normal data is hold-out as testing along with the attack dataset, and estimate the evaluation metrics between *Benign* or *Attack* classes. This research has experimented with the models with different time window lengths and reports the results for each time window since different values report different performance figures as shown in table 5.6 & table 5.7 for FTP Brute force and UDP DDoS attacks respectively. From the results, it is shown that the proposed model performance varied with time window length.

The detection performance of the proposed model is evaluated in terms of precision, recall, f1-score, and accuracy for various time windows (1 min, 2 min, 3 min, and 4 min). To estimate the prediction performance, 10% of normal data is used as testing along with the attack dataset, and estimated the evaluation metrics between *Benign* or *Attack* classes. This research work experimented the models with different time window lengths and report the results for each time window since different values exhibits different performance figures as mentioned in table 5.6 & table 5.7 for FTP Brute force and UDP DDoS attacks respectively. This research presents results for the two proposed methods, With majority voting and with self supervised learning separately, defined as Proposed M.V. and Proposed S.S. respectively. From the results, it is evident that the proposed model performance varied with time window

Table 5.6: *Performance Comparison of different auto-encoder models for FTP Bruteforce attack.*

Window Size $\triangle t$	**Model**	**Prec.%**	**Rec.%**	**F1.%**	**Acc.%**
1 min	Network View	93.0	100	96.3	94.1
	Flow View	94.9	100	97.3	95.8
	Image View	97.3	**100**	98.2	97.9
	Proposed M.V.	96.8	99.4	98.1	97.0
	Proposed S.S.	**98.7**	98.1	**98.3**	**98.5**
2 min	Network View	89.5	100	94.4	90.9
	Flow View	93.0	100	96.4	94.2
	Image View	95.6	100	**97.7**	95.9
	Proposed M.V.	93.9	98.9	96.4	94.2
	Proposed S.S.	**98.2**	96.0	95.9	**96.0**
3 min	Network View	92.4	98.3	95.3	92.5
	Flow View	91.0	98.3	94.5	91.2
	Image View	95.1	100	**97.5**	**95.9**
	Proposed M.V.	93.9	**100**	96.8	95.0
	Proposed S.S.	**98.3**	90.1	94.0	94.3
4 min	Network View	90.3	100	94.9	91.8
	Flow View	97.8	97.8	97.8	96.7
	Image View	97.6	100	97.6	96.0
	Proposed M.V.	96.0	**100**	**97.9**	95.7
	Proposed S.S.	**98.1**	93.1	95.5	**96.0**

Table 5.7: *Performance Comparison of different auto-encoder models for UDP DDoS attack.*

Window Size $\triangle t$	**Model**	**Prec.%**	**Rec.%**	**F1.%**	**Acc.%**
1 min	Network View	90.0	100	94.1	90.2
	Flow View	90.0	80	84.9	79.8
	Image View	93.0	97.5	95.2	93.6
	Proposed M.V.	93.0	**100**	**96.3**	**94.6**
	Proposed S.S.	**98.5**	77.12	86.5	90.8
2 min	Network View	87.2	100	93.1	89.6
	Flow View	91.6	80.4	85.7	81.0
	Image View	89.3	100	94.3	92.1
	Proposed M.V.	93.0	**100**	94.4	94.7
	Proposed S.S.	**97.9**	91.1	**95.4**	**95.5**
3 min	Network View	96.2	100	98.1	97.2
	Flow View	91.6	84.6	87.9	83.7
	Image View	87.0	100	93.1	90.4
	Proposed M.V.	96.2	**100**	**98.1**	**97.2**
	Proposed S.S.	**100**	81.0	90.0	93.2
4 min	Network View	84.3	100	91.3	86.2
	Flow View	90.0	95.2	93.0	89.6
	Image View	71.4	100	83.3	74.2
	Proposed M.V.	90.9	**100**	**95.2**	**92.8**
	Proposed S.S.	**100**	68.0	81.0	87.2

length. It is analyzed from the performance results that the proposed model outperforms every other individual view in terms of precision and accuracy for most of the time window values. The best f1-score and accuracy for FTP Brute force attack is attained as 98.3% and 98.5% respectively for a time window of 1 min. And The best f1-score and accuracy results for UDP DDoS attack is attained as 98.1% and 97.2% respectively for time window of 3 min.

It is also observed that the proposed majority voting based method always performs better than the flow view and network view separately for both the attacks. The proposed self-supervised learning-based model also outperforms the majority voting in terms of precision, f1-score and accuracy for specific time window length. It is estimated that recall values are lower for the proposed self supervised learning based model due to introduction of the slight change in threshold and generating data inherently which consists of more records for normal scenario , However with the proposed self supervised learning based scheme, model is able to capture the slight change in threshold values and handle the attack scenarios better.

It is also observed from view-specific results, that flow view and image view are able to well capture the FTP Brute Attacks because it is based on a connection-oriented protocol (i.e. TCP) and the performance results for network view are lower. On the other hand for UDP DDoS attack Network view performs better than flow view and image view because it is based on connection-less protocol (i.e. UDP) and network view is the best representative to explain the attack. It is also analyzed that image view performance (f1-score and accuracy) diminishes for larger window sizes.

5.5.3.2 Baseline Comparison

In this section, the performance evaluation of the proposed framework is discussed and the comparison with several baseline methods is presented for FTP brute force attack and UDP DDoS attack detection that are implemented using the CICIDS2018 dataset. The comparison results are presented in table 5.8 and table 5.9 for both the attacks. From the results it is evident that the proposed model outperforms most of these supervised learning based methods for both the attacks except [144] which is based on supervised learning based scheme and the classification results are better than proposed model.

Table 5.8: *Performance Comparison with earlier proposed methods for FTP Bruteforce attack.*

Method	Type	Prec.%	Rec.%	F1%	Acc.%
[145]	Supervised	82.8	42.6	56.3	42.6
[146]	Supervised	92.5	97.8	91.8	94.3
[147]	Supervised	95.7	95.4	97.5	-
[148]	Supervised	-	-	-	98.0
Proposed M.V.	Majority Voting	96.8	99.4	98.1	97.0
Proposed S.S.	Self-supervised	98.7	98.1	98.3	98.5

Table 5.9: *Performance Comparison with earlier proposed methods for UDP DDoS attack.*

Method	Type	Prec.%	Rec.%	F1%	Acc.%
[85]	Supervised	-	-	-	95
[149]	Supervised	88.0	100	94.11	93
[144]	Supervised	95.6	98.8	97.7	-
Proposed M.V.	Majority Voting	93.0	100	94.4	94.7
Proposed S.S.	Self-supervised	97.9	91.1	95.4	95.5

5.5.3.3 Time Window length vs Processing rate

Further, this section also investigates the processing rate in terms of records per second processed by the trained model against the provided time window length (in minutes), as shown in figure 5.5. The figure denotes that the processing rate is independent of selected time window as minimal change is identified on increasing time window length. This is due to non-contextual nature of the proposed scheme that process available data for every time window independently. However, the outcome of IDS is delayed due to higher time window length as IDS system requires to wait for processing till the data is available.

5.6 Summary

In this work, a novel approach is presented for attack detection that accommodate multiple types of views together. A multiview representation learning framework is proposed to incorporate multiple views together for attack detection. The proposed method relies on the network view, flow view, and image view of the captured network traffic for detecting the attacks. From the experimental results, it is evident that the proposed architecture outperforms the view-based models and chosen views are complimentary and useful to predict

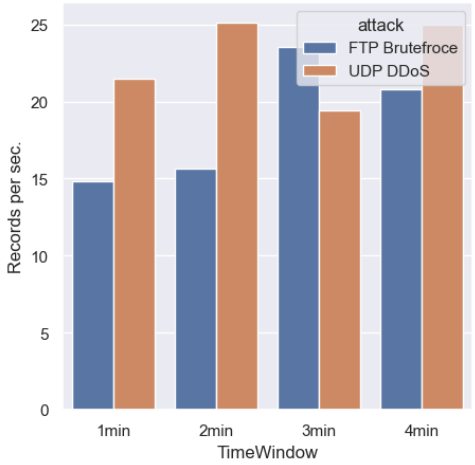

Figure 5.5: *Time window Length vs Processed records per sec.: CICIDS2018 Dataset*

the attacks robustly. It also proposed a self-supervised learning-based method for building an intrusion detection system by combining the multiple different views generated on raw packet data. The proposed method with the help of self-supervision automatically labels the input data and finally builds the classifier that can be used in the real network for attack detection. The proposed model also reduces the overhead of labeling every network packet for a specific attack and uses self-supervision with the majority voting of individually trained auto-encoders on a specific view. The implementation of intrusion detection is based on the latest & representative network packets available in the open-source community. From the results, it is evident that the proposed model significantly improves the results of combining the views together and outperforms the view-specific auto-encoder models.

6

Online Network Intrusion Detection System

6.1 Introduction

The previous chapters discuss the offline methods for building unsupervised learning-based methods for intrusion detection. In the first contribution 3, the offline training method is presented by utilizing the effect of weight decay in representation generation. In the second contribution 4, the protocol-related information is utilized in the attack detection process and offline training is performed for the proposed unsupervised learning-based method. A method to utilize the protocol-aware representations in the attack detection process is also discussed. In the last chapter 5, an approach utilizing the multiple views of the network communication information is presented and also introduced a self-supervised learning-based method for attack detection. In all of these earlier proposed methods, the model is trained offline on the training dataset and is used to perform prediction on the testing data. In offline learning-based methods, the model is trained usually once on the batch of data and continuously used in prediction, due to which it is unable to capture the changing patterns in data over time. For this purpose online learning-based method is utilized that can adapt to incoming traffic further. Online learning based methods [150] can be incremental or non-incremental in nature. Incremental learning based methods modify the trained model subsequently to consider new patterns for attack detection which is complex in nature. On

the other hand non-incremental methods learns the new patterns on-the-fly. This research work proposes an on-the-fly learning based method where model building and attack detection are performed for the data generated in a given time slot. With this method, the effect of change in data for every time slot can be integrated. Various experiments using flow-level information on a recent dataset is performed with varied time window sizes for FTP Brute Force and HTTP DDoS attacks respectively.

6.2 Need for an online detection system

With the rise of the Internet, attacks like DDoS, BruteForce, Botnet, etc. have become destructive and pose a great threat to overall security. The number of attacks on computer networks has been increased a lot over the years. These attacks can hamper the network performance, cause huge financial loss and overall credibility loss to an enterprise. Various open-source tools like LOIC [5], Slowloris [6], Nmap [7], Cain & Abel, etc can be used to perform different types of attacks easily. These attacks are performed against existing vulnerabilities in various networking protocols like SSH, FTP, HTTP, ICMP, etc. Each of these different attacks has a varied effect on network parameters. For example, Port scanning allows a lot of packets generated from one source port to multiple other destination ports that affect the distribution in destination ports. During high volume DDoS attacks sudden increase in packet transfer rate is identified. The major challenge is to identify these malicious activities with changing attack scenarios as model has to be updated with time. In recent years, several supervised intrusion detection systems have been proposed. However, these methods require labeled data for training and cannot automatically adapt to frequently changing network traffic scenarios. It is also required for data to be updated periodically and requires the model to be retrained to detect new attacks. This emphasizes the need for the development of unsupervised detection systems that can target zero-day attacks. Most of these supervised learning-based Intrusion Detection Systems (IDS) require huge manual effort for label generation and require data update over time to incorporate new attacks. It further emphasizes the need for an unsupervised learning method that can identify abnormal behavior automatically.

On the other hand for the continuous network traffic, it is expected that the model is able to learn the new patterns automatically which is not possible through offline systems. Since system behavior and applications are changing with respect to time, it is expected from the

model to update itself and adapt to a frequently changing environment so that it can be applied in real-time for attack detection. Hence it is required to build the IDS system that can be operated in an on-the-fly manner using an unsupervised machine learning method. This research proposed a solution that relies on identifying these anomalies in networking parameters over the sliding time window. The proposed system process data for each specific time window and detect the attack activity is present or not.

In this chapter, an unsupervised solution that relies on detecting attacks in a discrete-time sliding window using the distance between statistical features is proposed. The proposed algorithm utilizes generated cluster profiles and estimates the distance between statistical features to trigger an attack event if it exceeds the predefined threshold. The proposed solution does not rely on input from the last context and looks into only the current processing window for attack detection. The main contribution of the research can be summarized as follows:

- An unsupervised algorithm for network attack detection using clustering method is proposed that eliminates the need of labeled data.
- This research work proposed an on-the-fly learning based method for detecting attack independently for every time slot using flow-level information.
- The proposed system is based on extracting statistical features from the data and using specific distance threshold for attack detection.
- The recent benchmark dataset from Canadian Institute for Cybersecurity i.e. CICIDS2018 [1] is used to validate the proposed method.

The rest of the chapter is structured in the following way.Section 6.3 describes various machine learning-based methods for implementing online learning-based IDS solutions. Section 6.4 discusses the proposed algorithms to build online method for network attack detection using unsupervised mechanism. Section 6.5 describes various experimentation performed for detecting FTP Brute force and HTTP DDoS attacks. Lastly in section 6.6 the summary of the proposed approach is presented.

6.3 Related Work

This research study various machine learning-based online systems to detect network attacks. Most of the solutions devised are based on unsupervised and supervised learning.

6.3.1 Unsupervised learning based methods

Kitsune, [27] an ensemble of auto-encoders based detection system is proposed that can detect attacks without supervision. The proposed system is initially trained with normal traffic and later anomaly is identified based on reconstruction error. It consists of two parts an ensemble layer for training and an output layer for producing the final score of anomaly. It uses a total of 115 statistical features computed over 5 last time windows with damped incremental statistics. Feature reduction is performed using agglomerative hierarchical clustering. The major limitation is that it expects all traffic to be Benign in the training phase. Another ensemble framework is discussed in [151] which is using an ensemble of generative model with weighted average. The proposed model uses Mahalanobis distance to differentiate the anomaly from benign traffic. It expects Benign data available to train the ensemble models separately. However, training data can be poisoned with attack data which is similar to Normal traffic scenario.

ORUNADA system [54] proposes an online and scalable method for detecting network anomalies. The proposed system uses Incremental Density Grid-based clustering method with continuous update of feature space for discrete-time windows. It follows Evidence accumulation (EA4O) technique to identify anomalies. It uses aggregated features on source IP and destination IP like the number of packets, percentage of packets with different TCP flags set SYN/ACK/RST/FIN, percentage of different ICMP error related packets, average Packet Size, etc. Spanish ISP and MAWILab Dataset are used for testing the results.

Another online anomaly detection method for augmented network flows [152] is proposed that is based on count-min sketch for per-flow, per-node, per-network level statistics. It proposes SVM based adaptive mechanism and use dynamic input normalization approach for scaling, that also monitors the range and variance of observed values. Experimentation is performed on two datasets MAWI and Lawrence Berkeley National Laboratory. [153] discuss the method for detecting anomalies using traffic feature distributions. It uses entropy as a summarization tool to estimate the distributional changes in traffic features. Traffic flow consists of 4 tuples srcIP, dstIP, srcPort, dstPort. Sample Entropy is evaluated using feature distribution constructed from packet counts. It uses Hierarchical Agglomerative clustering for and proposes a multi-way subspace method for identifying anomalies across multiple traffic features and multiple origin-destination flows.

6.3.2 Supervised learning based methods

Supervised learning based solutions like DeepWindow [154] propose an online supervised learning-based method using LSTM for abnormal traffic detection using CICIDS2018 Dataset. It uses Mutual Information and Maximal information coefficient for feature selection. Features like count packets, count PSH flag, total time, packet max, average length, etc. are used with a window size of t periods. Another solution [155] discusses online method for DDoS attack detection. Being a supervised solution, it expects the data to be updated periodically to detect the new attack. In this section, major research in the area of Online attack detection for multiple different attacks have been explored. In the next section, the proposed solution and algorithm are discussed in detail.

The proposed method resolves these issues by introducing a on-the-fly learning based solution that is based on an unsupervised learning mechanism and process the results with lesser time complexity. In the next section, the proposed methodology is discussed in detail.

6.4 Proposed Methodology

6.4.1 Assumptions

1. The proposed method assumes that the flow level records can be generated with required features for specific attack detection.
2. The proposed method assumes that an individual model is built for specific attack only to distinguish between Normal and specific attack.

6.4.2 Architecture

The proposed solution for intrusion detection system consists of 3 major stages Data Preprocessing, Model Building & Attack Detection as specified in figure 6.1. All the stages are performed in sequence for every incoming flow records in Δt time period to identify malicious activity.

1. *Data Pre-processing*

   Data Pre-processing is an initial step that is required for any modeling task. It consists of two main process i.e. Feature Selection and Feature Scaling. Since flow-based

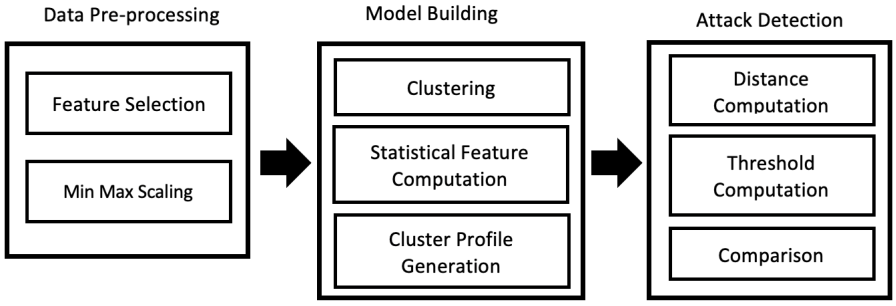

Figure 6.1: *High level architecture diagram for the proposed Online based IDS*

records contain a lot of parameters therefore It is needed to select those network parameters that are relevant for a specific attack. After proper analysis and using domain knowledge a subset of parameters is selected that can be used to identify the attack. The parameter set and its description for an individual attack is provided in table 6.1

Table 6.1: *Affected Network Parameters for specific attack*

Attack Type	**Parameter**
FTP Brute Force	Size of packet and flow records like Segment Size, average Packet size, etc.
Distributed Denial of Service	Rate of flow like Packets/s , Bytes/s etc.

Based on the parameter selection corresponding features are identified from the dataset. Once the feature is selected for specific attack detection it is required to scale the features in a definite range so that results should not get biased to larger values. For this, Min-Max Scaling technique is used where features are transformed in the range [0,1]. If feature values are represented in X then transformed value X_t is,

$$X_t = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{6.1}$$

2. *Model Building*

The main objective of this stage is to build the unsupervised model and generate the statistical features for attack detection. To reduce the search space it is required to combine similar points of records together into multiple groups. The proposed method performs the clustering for this purpose and generates optimum cluster by analyzing

the clustering quality using various evaluation measures like Purity, Homogeneity score, etc. After final clustering, it can be analyzed that similar flow-level information is combined together. The main objective of this step is to identify the abnormal cluster in later stages.

Once the clusters are created it is required to analyze the clusters and identify if any attack cluster exists with differentiating behavior. Profile Generation is executed to build the statistical insights from clustering which is an important step to compute the statistical features for every cluster. These statistical features are computed on each identified cluster and consist of various 1D and 2D features as described in table 6.2. Let F_i and F_j are two features with value set as X $<x_1, x_2, ...x_n>$and Y $<y_1, y_2, ...y_n>$.

After this process, each of the generated features V_i is named as $<S_i$-$F_i>$Where S_i represents a statistical parameter and F_i represents feature or group of features on which it is computed as based on statistical feature. Once the statistical features are computed cluster is represented in vector format as V which is called cluster profile as represented in table 6.3. In table 6.3 some of the computed cluster profile features are mentioned where FwdSegSizeMin represents the minimum segment size in the forward direction, FwdPkts/s and BwdPkts/s represent the packets per second transferred in the forward and backward direction. Profile generation is an important step that is used to map the clusters in vector representation which is further used to identify the attack.

Table 6.2: *List of statistical features used for online detection*

Dimension	**Feature**	**Formula**
1D	Mean	$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i$
	Variance	$\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2$
2D	Cosine Distance	$1 - \frac{X \cdot Y}{\lvert X\rvert\lvert Y\rvert}$
	Correlation Distance	$1 - \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \cdot \sigma_y}$

Table 6.3: *Sample of the generated cluster profiles for online detection method*

Cluster No.	Mean-FwdSegSize Min	Var-FwdSegSize Min	Mean-FwdPkts/s	Var-BwdPkts/s	Cor-FwdSegSizeMin-FwdPkts/s	Cosine-FwdSegSizeMin-FwdPkts/s
1	0.0	0.2	0.007	0.1	0.02	0.2
2	0.62	0.1	0.009	0.2	0.02	0.7

3. *Attack Detection*

   Attack Detection is the main stage to identify if any attack activity is present at the current time window. Once the cluster profiles are generated attack detection procedure follows to compute the inter-cluster distance among every pair of cluster. These distances are computed individually among created profile parameters and summed up together to compute the total distance as represented in the equation 6.2 where K is total number of generated statistical features and distance D_{ij} is computed between profile vectors V^i and V^j.

$$D_{ij} = \frac{1}{K} \sum_{k=1}^{K} |V_k^i - V_k^j|) \tag{6.2}$$

   The modulus value of the computed distance is considered because some of the features affect positively and others affect negatively in terms of statistical distance. For N different clusters total $(N^2 - N)$ different distances are computed among every pair of cluster. The maximum of the computed distance is compared with defined threshold θ. If the magnitude of distance is greater than θ attack is notified. Threshold value is adjusted based on administrator input on the outcome of the last N window results. It can also be updated based on requirement of reducing false positives or false negatives.

6.4.3 Algorithm

The proposed algorithm involves multiple stages of execution as described earlier. These stages impart a specific role in the overall process. Flow level records at a particular time window Δt and distance threshold θ are provided as input to the algorithm. On this subset of flow records, required features are selected and scaled at an early stage. Further clustering is performed to represent a group of similar points. At a later stage, statistical features are computed on each cluster to generate a cluster profile. Now objective is to identify the maximum distance attained among pairs of clusters. This maximum distance is matched

with a predefined distance threshold to identify the attack.

The overall complexity of the proposed algorithm is (O($Nlog(N)$) + O(C^2)) where N is number of data points in time slot t and C is the number of clusters created from data, if DBSCAN algorithm is used for clustering which has worst case complexity as O($Nlog(N)$). The whole process is explained as below in the Algorithm 3

Algorithm 3: Algorithm for Online Attack Detection

Input : Dataset at time window T as collection of flow level records with n attributes F_i where 1 $\leq$i$\leq$ n , Distance Threshold θ
Output: Attack or Benign

Select feature set F for specific attack identification.
F_{scaled} $\leftarrow$MIN-MAX-SCALING(F)
Clusters C $\leftarrow$CLUSTERING (F_{scaled})
Initialize Profile set as V
Select 1D and 2D statistical feature set as S
for $C_i \in C$ **do**
    Initialize Profile vector as V^i
    for $S_j \in S$ **do**
        V_i $\leftarrow$STATISTICAL-FEATURE-COMPUTATION (F_{scaled}, S_j)
    end for
    Append Profile $V^i \leftarrow V_i$
end for
Initialize distance array as D
for $V^i \in P$ **do**
    for $V^j \in P$ **do**
        D_{ij} $\leftarrow$DISTANCE-COMPUTATION (V^i, V^j)
        Add D_{ij} to D
    end for
end for
$D_{max} \leftarrow MAX - DISTANCE$(D)
if D_{max}>θ **then**
return Attack **else**
return Benign

We have also compared the time complexity of existing unsupervised learning based solutions in table 6.4 that are mentioned in background study 6.3.

Table 6.4: *Comparison of complexity with existing solutions*

Solution	Model Utilized	Time Complexity
Kitsune [27]	Ensemble of Autoencoder	O(N^2)
ORUNADA [54]	Density Grid based Clustering	O($Nlog(N)$)
Online adaptive [152]	Support Vector Machine	O(N^2)
Traffic Feature Distribution [153]	Hierarchical Agglomerative Clustering	O(N^3)
Proposed Algorithm	DBSCAN Clustering	O($Nlog(N)$)

6.5 Experimentation Results

6.5.1 DataSet

The CICIDS2018 Dataset [1] is used to train and test the proposed attack detection system. This is the recent dataset developed at the Canadian Institute of Cybersecurity in which records are collected over testbed with networking infrastructure containing 50 attacking machines and the victim organization has 5 departments and includes 420 machines and 30 servers. FTP-Patator and SSH-Patator tools [100] are used for performing brute force attacks and LOIC tool is used for DDoS attack. The generated Flows are captured using CICFlowMeter-V3 [102] utility. In total 7 different types of attacks are captured at different dates. The proposed method only selected the data for which FTP BruteForce attack and HTTP DDOS attack are performed along with Benign traffic as shown in the table 6.5.

Table 6.5: *Dataset used for online attack detection*

Attack Type	Class	Number of Records
FTP BruteForce	Benign	667626
	Attack	193360
HTTP-DDoS	Benign	7372557
	Attack	576191

6.5.2 Implementation

The implementation utilized flow level data captured on 14-02-2018 & 20-02-2018 specifically for FTP brute force attack and Http DDoS attack. The proposed method do not require

Table 6.6: *Attack specific features used for online detection*

Attack	Feature	Explanation
FTP BruteForce	Fwd Seg Size Min	Min Segment size in forward direction
	Fwd Pkts/s	Packets per sec in forward direction
	Bwd Pkts/s	Packets per sec in backward direction
	Pkt Size Avg	Average packet size
	Fwd Pkt Len Std	Standard Deviation of packet length in forward direction
	Bwd Pkt Len Std	Standard Deviation of packet length in backward direction
HTTP DDoS	Tot Fwd Pkts	Total packets in forward direction
	Tot Bwd Pkts	Total packets in backward direction
	TotLen Fwd Pkts	Total length of packets in forward direction
	TotLen Bwd Pkts	Total length of packets in backward direction
	Fwd IAT Tot	Total Inter-arrival time in forward direction
	Bwd IAT Tot	Total Inter-arrival time in backward direction
	Fwd Pkts/s	Packets per sec in forward direction
	Bwd Pkts/s	Packets per sec in backward direction

the associated label information to start the process. The complete implementation is done on Mac OS with 6 core intel core i7 processor with 16 GB RAM. Feature selection is performed based on attack property as described earlier. Table 6.6 provides the details on selected features for each type of attack detection. The proposed method used the DBSCAN Clustering algorithm [156] for the generation of clusters. This helps us to separate out the noise cluster which can be avoided in distance estimation. The python machine learning package i.e. scikit-learn [157] is used for the implementation.

6.5.3 Performance Metrics

The performance of the proposed algorithm is estimated in terms of clustering and classification. These metrics are computed for various window sizes.

6.5.3.1 Clustering Metrics

The proposed method uses labels from the data set for the computation of clustering measures. These measures also help us to select the best hyperparameters for clustering. Purity score and Homogeneity score are used to evaluate the quality of clusters formed as specified

Table 6.7: *Clustering results for both the attacks with varying time window size*

Attack	Window Size	Purity Score	Homogeneity Score
FTP BruteForce	2 min.	0.99	0.99
	1 min.	0.99	0.97
	30 sec.	0.98	0.96
	15 sec.	0.98	0.93
HTTP DDoS	2 min.	0.86	0.68
	1 min.	0.87	0.71
	30 sec.	0.89	0.73
	15 sec.	0.93	0.82

in section 2.6. The mean of the mentioned two scores is computed for a group of clusters identified at a time slot. In the proposed approach it is also required that each cluster has a single class of examples and a higher purity score.

6.5.3.2 Classification Metrics

To estimate the performance of the algorithm it is needed to estimate classification metrics in terms of precision, recall, f1-score, and accuracy metrics as specified in section 2.6.

6.5.4 Results and Analysis

6.5.4.1 Detection Performance

The detection performance of proposed method is evaluated for FTP Brute Force and Http DDoS Attack individually. As per CICIDS2018 dataset, FTP Brute Force attack is performed from 10:32 am till 12:09 pm on 14-02-2018. The data from 10:00 am till 1:00 pm time considering first 30 min and last 30 min as the Benign traffic is considered. Similarly another attack Http DDoS is performed on 20-02-2018 from 10:12 am till 11:17 am time and the data from 9:00 am till 12:30 pm is considered to include the benign traffic before and after attack. The results are estimated for different time window size of 2 min, 1 min, 30 sec, and 15 sec. DBSCAN algorithm [156] is utilized for clustering with configuration parameters eps = 0.005 and min-sample = 10 for both the attacks. Clustering evaluation metrics are attained for various window sizes and shown in table 6.7 for FTP BruteForce and HTTP DDoS attacks respectively.

To estimate the detection performance the time range is divided into multiple different

Table 6.8: *Classification results for both attacks with varying time window size*

Attack	Window Size	Precision	Recall	F1-score	Accuracy
FTP BruteForce	2 min.	97%	81%	88%	88%
	1 min.	98%	85%	91%	91%
	30 sec.	98%	93%	95%	95%
	15 sec.	95%	87%	91%	90%
HTTP DDoS	2 min.	90%	90%	90%	93%
	1 min.	91%	96%	93%	95%
	30 sec.	74%	91%	81%	85%
	15 sec.	54%	80%	64%	69%

slots based on window size and the results are estimated based on attack time duration. Overall classification metrics calculations are based on 2-class (Benign or Attack) classification results. Threshold value 2.5 is used for FTP Brute Force attack detection and value 47 is used for Http DDoS attack detection to estimate the detection performance. These respective thresholds are determined after analysis in sub-section 6.5.4.3. The attack detection results are represented in table 6.8 for both the attacks with varying window size.

6.5.4.2 Time Window Selection

Table 6.9: *Window Size vs Response Time*

Attack	Window Size	No. of Batches	Total Time(sec.)	Response Time(sec.)
FTP BruteForce	2 min.	89	43	0.48
	1 min.	177	44	0.24
	30 sec.	348	46	0.13
	15 sec.	674	59	0.08
HTTP DDoS	2 min.	91	884	9.71
	1 min.	181	524	2.89
	30 sec.	358	405	1.13
	15 sec.	693	404	0.58

A proper time window needs to be selected for processing the results in an estimated time (i.e. response time) depending upon the flow arrival rate. The proposed method estimated time taken for each batch to process for varying window size as shown in table 6.9 for detection of both the attacks individually. It is analyzed that response time is reduced on taking smaller window size and from the detection performance results, it is evident that reducing the window size after a certain point reduces the overall classification metrics. This

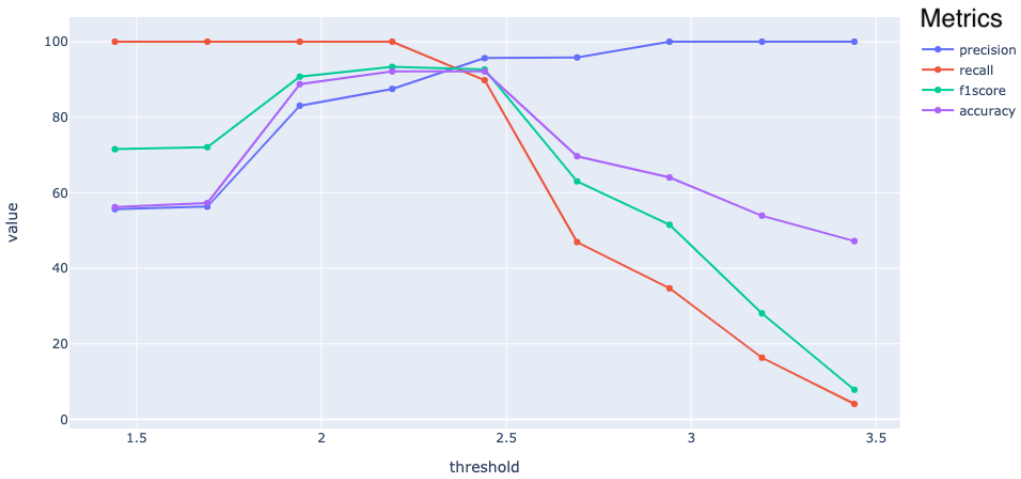

Figure 6.2: *Threshold Selection for FTP BruteForce attack*

can help to select the optimum window size based on flow arrival rate. For HTTP DDoS attack, processing time is larger than FTP Brute Force because of the higher volume of traffic generated during the DDoS attack.

6.5.4.3 Threshold Selection

Threshold plays a vital role in getting the good results. To decide the threshold it is needed to check the performance metrics at various thresholds. The minimum and maximum distance achieved for a specific time window size of 2 min is used and the metrics are estimated. The proposed method selects the threshold which results on best performance metrics identified as 2.5 for FTP Brute Force and 47 for HTTP DDoS attacks as shown in figure 6.2 & 6.3.

6.6 Summary

This research presented an on-the-fly learning based approach for attack detection that works in an unsupervised manner. The proposed system reduces the overhead of labeling every network flow data. The administrator can decide the threshold based on networking scenarios and the kind of attack that needs to be detected. The implementation of intrusion detection is based on the latest and representative flow level dataset. The current approach

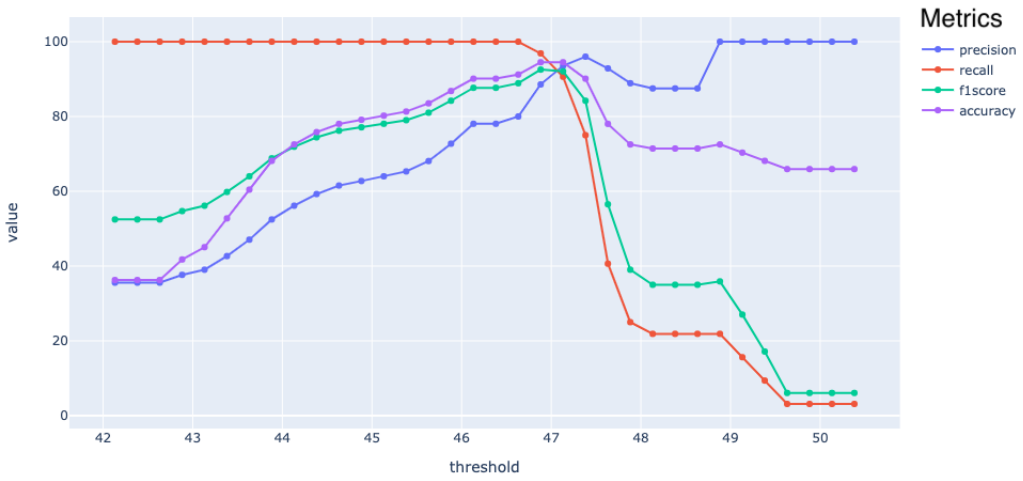

Figure 6.3: *Threshold Selection for HTTP DDoS attack*

does not require the previous context of time windows that allows it to execute independently for each time step.

7

Conclusions and Future Perspectives

This chapter presents the summary of the work, highlights the contributions, and suggests the directions for possible future work.

7.1 Summary of the contribution of thesis

This thesis has proposed various unsupervised learning-based methods for Network Intrusion detection systems. The summary of the various methods presented and the chapter-wise contributions are as follows:

In **Chapter 3**, the influence of temporal context in building an unsupervised learning-based network intrusion detection system is discussed. The concept of regeneration error is utilized to build the autoencoder-based model by considering the previous context with a weighted decay factor using a multi-layer Time aware LSTM network. It involves regularizing the feature and regenerating the input data using an autoencoder-based model. It is analyzed that consideration of the sequence of records over consequent timestamps played an important role in determining the attack activity due to the sequential nature of the attack life-cycle. The proposed system is based on an unsupervised setup to estimate reconstruction error and identify attacks based on a threshold obtained only from normal traffic. It is also inferred that Time time-aware LSTM-based autoencoder is able

to capture time-dependent contextual information and comparison with baseline methods of MLP and LSTM-based autoencoder justifies that diminishing context over time plays an important role in accurate attack identification. Multiple recent benchmark datasets - CICIDS2018, and CICDDoS2019 are used for evaluating the performance of the suggested method for FTP BruteForce and UDPDoS attacks respectively. It is also observed that the performance of the standalone OCSVM-based model is improved further after utilizing the intermediate representations from the autoencoder models. It is also analyzed that the TLSTM-based OCSVM model is performing better than the LSTM-based OCSVM model for all the data sets.

In **Chapter 4**, the importance of protocol awareness for generating a better representation in an unsupervised learning-based mechanism is discussed. To incorporate protocol-aware information while modeling, a method for protocol-aware unsupervised network intrusion detection systems is proposed that can learn the protocol-aware representations automatically and utilize this information for the final decision. It is described that the data received by a device at a particular instance of time may consist of heterogeneous information related to different protocols and anomaly characteristics of different protocol-specific attacks may also be different. Therefore capturing the differences in characteristics of different protocols may be important to effectively determine anomalies present in data. However, a generalized global representation may fail to effectively capture heterogeneous characteristics of different protocols, which may be important for anomaly detection. The proposed unsupervised learning-based method not only generates a global representation of the entire data but also generates local representation for different protocols and learns the importance of local representation in attack determination by utilizing a self-attention-based network. It is concluded that the protocol-specific unsupervised autoencoder-based model is able to capture time-dependent contextual information and identify specific attacks at the protocol level. The proposed system is based on an unsupervised setup to estimate reconstruction error and identify attacks based on a threshold obtained from normal traffic. A method to combine protocol-specific encoders with an attention network is also presented that utilizes the importance of each protocol channel for attack identification and helps in improving the results further. Multiple recent benchmark datasets - CICIDS2018 and CICDDoS2019 are used for evaluating the performance of the proposed model on various attacks like Brute-Force (FTP Bruteforce and SSH Bruteforce) and DDoS attacks (SYN Flood, UDPDoS, and NTP DoS) respectively, and from the results, it is observed that

the proposed model outperforms the model created for full data for both CICIDS2018 and CICDDoS2019 datasets.

In **Chapter 5**, multiple views from network packets are utilized to build an unsupervised learning-based model. The importance of each different view in identifying the network attacks is presented, which can help in improving the overall detection performance by learning differentiated features from the multiple views together. Further, the view-specific autoencoders are built on extracted view-specific features and proposed majority voting and self-supervised learning-based models for attack detection. The majority voting-based method is used to consider the output from view-specific autoencoders and the final outcome is based on the majority of the individual autoencoder's decision. On the other hand, a self-supervised learning-based model is discussed that is based on majority voting results and considers the minimal change in estimated thresholds. The proposed self-supervised learning approach utilizes the reconstruction error from each view and labels the incoming data with a majority voting of individual view-specific encoders. This inherent labeled dataset is finally used to build the binary classifier and utilize it in attack detection. The recent benchmark dataset i.e. CICIDS2018 is used for evaluating the performance of the proposed model on FTP brute force and UDP DDoS attacks and results are compared for view-specific auto-encoder. The experiments are analyzed with different time window lengths of 1 min, 2 min, 3 min and 4 min. and it is concluded that considering multiple views together improves the overall metrics.

Finally, in **Chapter 6**, on-the-fly learning based method for an intrusion detection system is presented that relies on identifying these anomalies in flow-level network data over the sliding time window. The proposed system processes data for each specific time window and detects whether the attack activity is present or not. The proposed solution does not rely on input from the last context and looks into only the data in the current processing window for attack detection. The proposed unsupervised learning-based solution relies on detecting attacks in discrete time sliding windows that first perform clustering on the extracted subset of data to group similar records together and further create the cluster profiles by computing the 1D and 2D statistical features on flow-level data. These cluster profiles are ultimately used to estimate the statistical distance between computed clusters and trigger an attack if it exceeds the predefined threshold. The recent benchmark dataset from the Canadian Institute for Cybersecurity i.e. CICIDS2018 is used to validate the proposed method for FTP Brute Force and HTTPDDoS attacks. The experimental

results are discussed for computing detection performance in terms of f1-score and accuracy primarily. From experimentation results it is observed that the f1-score increases on reducing the window size up to a point and reduces thereafter which helps in determining the suitable window size. Further attack-specific threshold values are also estimated by looking into the metrics and estimating minimum and maximum computed distance for various threshold values.

7.2 Future Perspectives

The work reported in the chapters of this thesis provide ample scope and promulgate several clear directions for future research endeavors. The following are some possible research directions.

- The proposed multiview-based unsupervised mechanism ignores the previous context information while making the decision. Thus the approach can be extended to support the contextual information by utilizing the LSTM-based autoencoder models.
- As different thresholds are required to be learned while training the model and before deploying it to production, the effort can be given to developing the scheme to provide an adaptive threshold mechanism.
- The proposed online scheme to detect attacks can be extended to work on aggregated flows that can help reduce the complexity of processing every network flow record. The proposed online method can also be extended to utilize the contextual information to improve the results further. The proposed methods can further be extended to support incremental learning that can avoid training the model for every incoming data at a particular time window.
- As IoT networks are getting popular, and the major challenge is to build a mechanism that consumes fewer resources while inference. The applicability of the proposed unsupervised learning based solutions may be studied in this domain.

Publications

1. **Ritesh Ratti**, Sanasam Ranbir Singh, Sukumar Nandi "Multiview-based network Intrusion detection system", submitted to journal Transactions on Dependable and Secure Computing, (Under Review).
2. **Ritesh Ratti**, Sanasam Ranbir Singh, Sukumar Nandi "Protocol aware unsupervised network intrusion detection system", ITCCN symposium of the 22nd IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom-2023), Exeter, U.K.
3. **Ritesh Ratti**, Sanasam Ranbir Singh, Sukumar Nandi "Network Intrusion detection system using Time aware LSTM network" 21st IEEE International Conferences on Embedded and Ubiquitous Computing (IEEE EUC 2023), Exeter, U.K.
4. **Ritesh Ratti**, Sanasam Ranbir Singh, Sukumar Nandi, "Online network intrusion detection system using statistical features" 2021 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), India, Pages 125-130.
5. **Ritesh Ratti**, Sanasam Ranbir Singh, Sukumar Nandi, "Towards implementing fast and scalable network intrusion detection system using entropy based discretization technique," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) India, 1-7.

References

[1] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018. [Pg.xvi], [Pg.xviii], [Pg.34], [Pg.35], [Pg.40], [Pg.48], [Pg.61], [Pg.72], [Pg.79], [Pg.80], [Pg.93], [Pg.103], [Pg.110]

[2] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8. [Pg.xvi], [Pg.xviii], [Pg.35], [Pg.36], [Pg.40], [Pg.49], [Pg.61], [Pg.72]

[3] (2021) https://securityboulevard.com/2021/03/cybercrime-to-cost-over-10-trillion-by-2025/ . [Pg.1]

[4] (2020) Mcafee report https://www.mcafee.com/de-ch/consumer-corporate/newsroom/press-releases/press-release.html?news_id=6859bd8c-9304-4147-bdab-32b35457e629. [Pg.1]

[5] M. Sauter, ""LOIC Will Tear Us Apart": The Impact of Tool Design and Media Portrayals in the Success of Activist DDOS Attacks," *American Behavioral Scientist*, vol. 57, no. 7, pp. 983–1007, 2013. [Pg.1], [Pg.60], [Pg.78], [Pg.102]

[6] G. Yaltirakli, "Slowloris," *https://github.com/gkbrk/slowloris*, 2015. [Pg.1], [Pg.60], [Pg.78], [Pg.102]

[7] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US), 2008. [Pg.1], [Pg.60], [Pg.78], [Pg.102]

[8] G. H. Kim and E. H. Spafford, "The design and implementation of tripwire: A file system integrity checker," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, 1994, pp. 18–29. [Pg.3]

[9] B. Wotring, *Host integrity monitoring using Osiris and Samhain.* Elsevier, 2005. [Pg.3]

[10] A. Hay, D. Cid, and R. Bray, *OSSEC Host-Based Intrusion Detection Guide.* Syngress Publishing, 2008. [Pg.3]

[11] M. Ahmed, Abdun, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, 2016. [Pg.3]

[12] J. Zhang, M. Zulkernine, and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008. [Pg.3]

[13] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A Survey of Deep Learning-Based Network Anomaly Detection," *Cluster Computing*, vol. 22, no. 1, p. 949–961, jan 2019. [Pg.3]

[14] K. A. Taher, B. Mohammed Yasin Jisan, and M. M. Rahman, "Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection," in *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, 2019, pp. 643–646. [Pg.3], [Pg.7], [Pg.60]

[15] F. Gharibian and A. A. Ghorbani, "Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection," in *Fifth Annual Conference on Communication Networks and Services Research (CNSR '07)*, 2007, pp. 350–358. [Pg.3]

[16] S. Kaddoura, A. E. Arid, and M. Moukhtar, "Evaluation of supervised machine learning algorithms for multi-class intrusion detection systems," in *Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3*, K. Arai, Ed. Cham: Springer International Publishing, 2022, pp. 1–16. [Pg.3], [Pg.7], [Pg.60]

[17] P. Ioulianou, V. Vasilakis, I. Moscholios, and M. Logothetis, "A signature-based intrusion detection system for the internet of things," *Information and Communication Technology Form*, 2018. [Pg.3]

[18] F. Erlacher and F. Dressler, "On high-speed flow-based intrusion detection using snort-compatible signatures," *IEEE Transactions on Dependable and Secure Computing*, 2020. [Pg.3]

[19] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to Snort system," *Future Generation Computer Systems*, vol. 80, pp. 157–170, 2018. [Pg.3]

[20] N. M. . N. Besharati, E., "LR-HIDS: logistic regression host-based intrusion detection system for cloud environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, p. 3669–3692, 2019. [Pg.3], [Pg.19]

[21] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238. [Pg.3], [Pg.78]

[22] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999. [Pg.3], [Pg.78]

[23] V. Jyothsna, R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011. [Pg.4], [Pg.61]

[24] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers Security*, vol. 116, p. 102675, 2022. [Pg.4], [Pg.61]

[25] K. Flanagan, E. Fallon, P. Connolly, and A. Awad, "Network anomaly detection in time series using distance based outlier detection with cluster density analysis," in *2017 Internet Technologies and Applications (ITA)*. IEEE, 2017, pp. 116–121. [Pg.4]

[26] P. Bereziński, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367–2408, 2015. [Pg.4]

[27] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018. [Pg.4], [Pg.6], [Pg.8], [Pg.9], [Pg.28], [Pg.41], [Pg.44], [Pg.63], [Pg.64], [Pg.81], [Pg.104], [Pg.110]

[28] 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. [Pg.4], [Pg.8], [Pg.28], [Pg.41], [Pg.44], [Pg.54], [Pg.63], [Pg.64], [Pg.81]

[29] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, p. 102675, 2022. [Pg.5]

[30] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016. [Pg.5]

[31] Y. Wang, J. Wong, and A. Miner, "Anomaly intrusion detection using one class svm," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, 2004, pp. 358–364. [Pg.5], [Pg.8]

[32] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proceedings of the conference on research in adaptive and convergent systems*, 2019, pp. 161–168. [Pg.5], [Pg.21]

[33] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41 238–41 248, 2018. [Pg.6], [Pg.29]

[34] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018. [Pg.6], [Pg.29]

[35] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne)," *Computer Science Review*, vol. 40, p. 100378, 2021. [Pg.6]

[36] D. Li, D. Kotani, and Y. Okabe, "Improving attack detection performance in nids using gan," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC).* IEEE, 2020, pp. 817–825. [Pg.6]

[37] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14.* Springer, 2019, pp. 622–637. [Pg.6]

[38] W. Xu, H. Sun, C. Deng, and Y. Tan, "Variational autoencoder for semi-supervised text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017. [Pg.6]

[39] A. Gogna and A. Majumdar, "Semi supervised autoencoder," in *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part II 23*. Springer, 2016, pp. 82–89. [Pg.6]

[40] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and reliability engineering international*, vol. 17, no. 2, pp. 105–112, 2001. [Pg.6], [Pg.22]

[41] C. Krügel, T. Toth, and E. Kirda, "Service specific anomaly detection for network intrusion detection," in *Proceedings of the 2002 ACM symposium on Applied computing*, 2002, pp. 201–208. [Pg.6], [Pg.22]

[42] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: poster and demo track*, vol. 1, pp. 59–63, 2012. [Pg.6], [Pg.21]

[43] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, and C. D. Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert systems with Applications*, vol. 38, no. 1, pp. 306–313, 2011. [Pg.6], [Pg.20]

[44] N. H. Son and H. T. Dung, "The method of detecting online password attacks based on high-level protocol analysis and clustering techniques," *arXiv preprint arXiv:1912.02036*, 2019. [Pg.6], [Pg.20]

[45] H. Gwon, C. Lee, R. Keum, and H. Choi, "Network Intrusion Detection based on LSTM and Feature Embedding," *CoRR*, 2019. [Pg.7], [Pg.60]

[46] R. Singh, H. Kumar, and R. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609–8624, 2015. [Pg.7], [Pg.60], [Pg.62]

[47] S. Gupta, P. Kumar, and A. Abraham, "A profile based network intrusion detection and prevention system for securing cloud environment," *International Journal of Distributed Sensor Networks*, vol. 9, no. 3, p. 364575, 2013. [Pg.7], [Pg.60], [Pg.62]

[48] R. Patil, H. Dudeja, S. Gawade, and C. Modi, "Protocol Specific Multi-Threaded Network Intrusion Detection System (PM-NIDS) for DoS/DDoS Attack Detection in Cloud," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2018, pp. 1–7. [Pg.7], [Pg.62]

[49] M. Li, D. Han, X. Yin, H. Liu, and D. Li, "Design and implementation of an anomaly network traffic detection model integrating temporal and spatial features," *Security and Communication Networks*, vol. 2021, pp. 1–15, 2021. [Pg.8], [Pg.82]

[50] M. A. Khan, "HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, p. 834, 2021. [Pg.8], [Pg.43], [Pg.82]

[51] X. Gu, L. Akoglu, and A. Rinaldo, "Statistical analysis of nearest neighbor methods for anomaly detection," 2019. [Pg.8]

[52] N. Paulauskas and A. F. Bagdonas, "Local outlier factor use for the network flow anomaly detection," *Security and Communication Networks*, vol. 8, no. 18, pp. 4203–4212, 2015. [Pg.8]

[53] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft, "In-network pca and anomaly detection," *Advances in neural information processing systems*, vol. 19, 2006. [Pg.8]

[54] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2016. [Pg.9], [Pg.104], [Pg.110]

[55] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980. [Pg.13]

[56] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987. [Pg.13]

[57] B. A. Forouzan, *TCP/IP protocol suite*. McGraw-Hill Higher Education, 2002. [Pg.14], [Pg.59]

[58] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *International Journal of Advanced Networking and Applications*, vol. 7, no. 4, p. 2828, 2016. [Pg.18]

[59] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira'H, and N. Ababneh, "An intelligent tree-based intrusion detection model for cyber security," *Journal of Network and Systems Management*, vol. 29, pp. 1–18, 2021. [Pg.18]

[60] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008. [Pg.18]

[61] W. Zong, Y.-W. Chow, and W. Susilo, "A two-stage classifier approach for network intrusion detection," in *Information Security Practice and Experience: 14th International Conference, ISPEC 2018, Tokyo, Japan, September 25-27, 2018, Proceedings 14*. Springer, 2018, pp. 329–340. [Pg.18]

[62] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *Ieee Access*, vol. 7, pp. 82 512–82 521, 2019. [Pg.18]

[63] M. Mohammadi, T. A. Rashid, S. H. T. Karim, A. H. M. Aldalwie, Q. T. Tho, M. Bidaki, A. M. Rahmani, and M. Hosseinzadeh, "A comprehensive survey and taxonomy of the svm-based intrusion detection systems," *Journal of Network and Computer Applications*, vol. 178, p. 102983, 2021. [Pg.18]

[64] D. Jing and H.-B. Chen, "Svm based network intrusion detection for the unsw-nb15 dataset," in *2019 IEEE 13th international conference on ASIC (ASICON)*. IEEE, 2019, pp. 1–4. [Pg.18]

[65] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "Svm-dt-based adaptive and collaborative intrusion detection," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 108–118, 2017. [Pg.18]

[66] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119–128, 2012. [Pg.18]

[67] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden naïve bayes multiclass classifier," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13 492–13 500, 2012. [Pg.18], [Pg.78]

[68] F. Jemili, M. Zaghdoud, and M. B. Ahmed, "A framework for an adaptive intrusion detection system using bayesian network," in *2007 IEEE Intelligence and Security Informatics*. IEEE, 2007, pp. 66–70. [Pg.18]

[69] A. Cemerlic, L. Yang, and J. M. Kizza, "Network intrusion detection based on bayesian networks." in *SEKE*, 2008, pp. 791–794. [Pg.18]

[70] M. Mehdi, S. Zair, A. Anou, and M. Bensebti, "A bayesian networks in intrusion detection systems," *Journal of computer Science*, vol. 3, no. 5, pp. 259–265, 2007. [Pg.18]

[71] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *arXiv preprint arXiv:1701.02145*, 2017. [Pg.19]

[72] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1222–1228. [Pg.19]

[73] Y. Wang, "A multinomial logistic regression modeling approach for anomaly intrusion detection," *Computers & Security*, vol. 24, no. 8, pp. 662–674, 2005. [Pg.19]

[74] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised clustering approach for network anomaly detection," in *Networked Digital Technologies: 4th International Conference, NDT 2012, Dubai, UAE, April 24-26, 2012. Proceedings, Part I 4*. Springer, 2012, pp. 135–145. [Pg.19]

[75] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-based systems*, vol. 78, pp. 13–21, 2015. [Pg.20]

[76] R. Hofstede, M. Jonker, A. Sperotto, and A. Pras, "Flow-based web application brute-force attack and compromise detection," *Journal of network and systems management*, vol. 25, pp. 735–758, 2017. [Pg.20]

[77] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *International workshop on recent advances in intrusion detection*. Springer, 2004, pp. 203–222. [Pg.21]

[78] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "Mcpad: A multiple classifier system for accurate payload-based anomaly detection," *Computer networks*, vol. 53, no. 6, pp. 864–881, 2009. [Pg.21], [Pg.78]

[79] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, 2002. [Pg.22]

[80] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 2, pp. 447–456, 2013. [Pg.22]

[81] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. [Pg.23]

[82] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006. [Pg.24]

[83] G. Li, P. Niu, X. Duan, and X. Zhang, "Fast learning network: a novel artificial neural network with a fast learning speed," *Neural Computing and Applications*, vol. 24, pp. 1683–1695, 2014. [Pg.24]

[84] 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. [Pg.24]

[85] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020. [Pg.25], [Pg.99]

[86] X. Zhang, J. Chen, Y. Zhou, L. Han, and J. Lin, "A multiple-layer representation learning model for network-based attack detection," *IEEE Access*, vol. 7, pp. 91 992–92 008, 2019. [Pg.25]

[87] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE access*, vol. 8, pp. 32 464–32 476, 2020. [Pg.25]

[88] 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. [Pg.25]

[89] 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. [Pg.26]

[90] 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. [Pg.27]

[91] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48 697–48 707, 2018. [Pg.27]

[92] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Pg.27]

[93] P. Malhotra, L. Vig, G. Shroff, P. Agarwal *et al.*, "Long short term memory networks for anomaly detection in time series." in *Esann*, vol. 2015, 2015, p. 89. [Pg.28]

[94] L. Bontemps, V. L. Cao, J. McDermott, and N.-A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Future Data and Security Engineering: Third International Conference, FDSE 2016, Can Tho City, Vietnam, November 23-25, 2016, Proceedings 3*. Springer, 2016, pp. 141–152. [Pg.28]

[95] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. [Pg.30], [Pg.31], [Pg.61], [Pg.69], [Pg.70]

[96] M. Tan, A. Iacovazzi, N.-M. M. Cheung, and Y. Elovici, "A neural attention model for real-time network intrusion detection," in *2019 IEEE 44th conference on local computer networks (LCN)*. IEEE, 2019, pp. 291–299. [Pg.31]

[97] F. W. L. W. P. A. Stolfo, Salvatore and P. Chan, "KDD Cup 1999 Data," UCI Machine Learning Repository, 1999, DOI: https://doi.org/10.24432/C51C7N. [Pg.32]

[98] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. [Pg.33]

[99] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. [Pg.33]

[100] Patator. FTP Patator and SSH Patator https://github.com/lanjelot/patator. [Pg.35], [Pg.93], [Pg.110]

[101] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *2016 International Conference on Information Science and Security (ICISS)*. IEEE, 2016, pp. 1–6. [Pg.36]

[102] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISSp*, 2017, pp. 253–262. [Pg.36], [Pg.42], [Pg.60], [Pg.72], [Pg.86], [Pg.93], [Pg.94], [Pg.110]

[103] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, "Patient Subtyping via Time-Aware LSTM Networks," ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–74. [Pg.41], [Pg.42], [Pg.49]

[104] Y. Zhang, "ATTAIN: Attention-based Time-Aware LSTM Networks for Disease Progression Modeling." in *In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-2019), pp. 4369-4375, Macao, China.*, 2019. [Pg.42]

[105] A. Nguyen, S. Chatterjee, S. Weinzierl, L. Schwinn, M. Matzner, and B. Eskofier, "Time matters: time-aware lstms for predictive business process monitoring," in *International Conference on Process Mining*. Springer, 2020, pp. 112–123. [Pg.42]

[106] R. Sommer and A. Feldmann, "NetFlow: Information Loss or Win?" in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 173–174. [Pg.42], [Pg.60], [Pg.86]

[107] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020. [Pg.42]

[108] 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. [Pg.42]

[109] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, p. 105124, 2020. [Pg.42]

[110] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 2018, pp. 178–183. [Pg.42]

[111] Z. Shi, J. Li, C. Wu, and J. Li, "DeepWindow: An Efficient Method for Online Network Traffic Anomaly Detection," in *2019 IEEE 21st International Conference on High Performance Computing and Communications*, 2019, pp. 2403–2408. [Pg.43]

[112] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE Access*, vol. 8, pp. 165 263–165 284, 2020. [Pg.44]

[113] A. Binbusayyis and T. Vaiyapuri, "Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM," *Applied Intelligence*, vol. 51, no. 10, pp. 7094–7108, 2021. [Pg.44], [Pg.63], [Pg.64], [Pg.81]

[114] X. Han, Y. Liu, Z. Zhang, X. Lü, and Y. Li, "Sparse auto-encoder combined with kernel for network attack detection," *Computer Communications*, vol. 173, pp. 14–20, 2021. [Pg.44], [Pg.63], [Pg.81]

[115] S. Harush, Y. Meidan, and A. Shabtai, *DeepStream: Autoencoder-Based Stream Temporal Clustering*. New York, NY, USA: Association for Computing Machinery, 2021, p. 445–448. [Pg.44]

[116] H. Neuschmied, M. Winter, K. Hofer-Schmitz, B. Stojanovic, and U. Kleb, "Two Stage Anomaly Detection for Network Intrusion Detection." in *ICISSP*, 2021, pp. 450–457. [Pg.44]

[117] G. Andresini, A. Appice, and D. Malerba, "Autoencoder-based deep metric learning for network intrusion detection," *Information Sciences*, 2021. [Pg.45], [Pg.63], [Pg.81]

[118] H. Hindy, C. Tachtatzis, R. Atkinson, D. Brosset, M. Bures, I. Andonovic, C. Michie, and X. Bellekens, "Leveraging Siamese networks for One-Shot intrusion detection model," *arXiv preprint arXiv:2006.15343*, 2020. [Pg.45], [Pg.63], [Pg.81]

[119] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, 2013, pp. 8–15. [Pg.54]

[120] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 488–498. [Pg.54]

[121] J. R. Rose, M. Swann, G. Bendiab, S. Shiaeles, and N. Kolokotronis, "Intrusion detection using network traffic profiling and machine learning for IoT," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 409–415. [Pg.62], [Pg.82]

[122] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the*

IEEE/CVF international conference on computer vision, 2019, pp. 1314–1324. [Pg.63], [Pg.82]

[123] M. Zolanvari, A. Ghubaish, and R. Jain, "ADDAI: Anomaly Detection using Distributed AI," in *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, vol. 1, 2021, pp. 1–6. [Pg.63]

[124] M. Labonne, A. Olivereau, B. Polvé, and D. Zeghlache, "Unsupervised Protocol-based Intrusion Detection for Real-world Networks," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 299–303. [Pg.63]

[125] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li, "Opentag: Open attribute value extraction from product profiles," in *Proceedings of the 24th ACM SIGKDD Conference*, 2018, pp. 1049–1058. [Pg.69], [Pg.70], [Pg.72]

[126] E. Papadogiannaki, G. Tsirantonakis, and S. Ioannidis, "Network intrusion detection in encrypted traffic," in *2022 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2022, pp. 1–8. [Pg.78]

[127] C. Guo, Y. Ping, N. Liu, and S.-S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, pp. 391–400, 2016. [Pg.78]

[128] P. Bhale, D. R. Chowdhury, S. Biswas, and S. Nandi, "Optimist: Lightweight and transparent ids with optimum placement strategy to mitigate mixed-rate ddos attacks in iot networks," *IEEE Internet of Things Journal*, 2023. [Pg.78]

[129] D. Ariu, R. Tronci, and G. Giacinto, "Hmmpayl: An intrusion detection system based on hidden markov models," *computers & security*, vol. 30, no. 4, pp. 221–241, 2011. [Pg.78]

[130] S. A. Thorat, A. K. Khandelwal, B. Bruhadeshwar, and K. Kishore, "Payload content based network anomaly detection," in *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*. IEEE, 2008, pp. 127–132. [Pg.78]

[131] H. V. Jagadish, "Analysis of the Hilbert curve for representing two-dimensional space," *Information Processing Letters*, vol. 62, no. 1, pp. 17–22, 1997. [Pg.80], [Pg.81], [Pg.86]

[132] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, 2018. [Pg.80]

[133] P. K. Chan and M. V. Mahoney, "PHAD: Packet header anomaly detection for identifying hostile network traffic," 2001. [Pg.80]

[134] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, no. 1-2, pp. 105–136, 2002. [Pg.80]

[135] S. S. Kim and A. N. Reddy, "Statistical techniques for detecting traffic anomalies through packet header data," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 562–575, 2008. [Pg.81]

[136] R. L. Tomio, E. K. Viegas, A. O. Santin, and R. R. dos Santos, "A Multi-View Intrusion Detection Model for Reliable and Autonomous Model Updates," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6. [Pg.81]

[137] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International COnference*, 2010, pp. 1–12. [Pg.81]

[138] I. Baptista, S. Shiaeles, and N. Kolokotronis, "A novel malware detection system based on machine learning and binary visualization," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–6. [Pg.81]

[139] F. Shen and O. Hasegawa, "Self-organizing incremental neural network and its application," in *Artificial Neural Networks–ICANN 2010: 20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III 20*. Springer, 2010, pp. 535–540. [Pg.81]

[140] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, and N. Kolokotronis, "Malware Squid: A Novel IoT Malware Traffic Analysis Framework using Convolutional Neural Network and Binary Visualisation," *CoRR*, vol. abs/2109.03375, 2021. [Online]. Available: https://arxiv.org/abs/2109.03375 [Pg.81]

[141] A. Cortesi, "binvis. io: Visual Analysis of Binary Files," 2016. [Pg.81], [Pg.86], [Pg.94]

[142] P. Goyal and A. Goyal, "Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark," in *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2017, pp. 77–81. [Pg.83]

[143] "Editcap." [Online]. Available: https://www.wireshark.org/docs/man-pages/editcap.html [Pg.94]

[144] I. O. Lopes, D. Zou, I. H. Abdulqadder, F. A. Ruambo, B. Yuan, and H. Jin, "Effective network intrusion detection via representation learning: A denoising autoencoder approach," *Computer Communications*, vol. 194, pp. 55–65, 2022. [Pg.98], [Pg.99]

[145] G. Kaur, A. H. Lashkari, and A. Rahali, "Intrusion traffic detection and characterization using deep image learning," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2020, pp. 55–62. [Pg.99]

[146] S. K. Wanjau, G. M. Wambugu, and G. N. Kamau, "Ssh-brute force attack detection model based on deep learning," 2021. [Pg.99]

[147] T. H. Hai and L. H. Nam, "A practical comparison of deep learning methods for network intrusion detection," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE, 2021, pp. 1–6. [Pg.99]

[148] J. Kim, Y. Shin, E. Choi *et al.*, "An intrusion detection model based on a convolutional neural network," *Journal of Multimedia Information System*, vol. 6, no. 4, pp. 165–172, 2019. [Pg.99]

[149] Q.-y. Dai, B. Zhang, S.-q. Dong *et al.*, "A ddos-attack detection method oriented to the blockchain network layer," *Security and Communication Networks*, vol. 2022, 2022. [Pg.99]

[150] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021. [Pg.101]

[151] J. Liao, S. G. Teo, P. P. Kundu, and T. Truong-Huu, "ENAD: An Ensemble Framework for Unsupervised Network Anomaly Detection." [Pg.104]

[152] D. Ippoliti, C. Jiang, Z. Ding, and X. Zhou, "Online adaptive anomaly detection for augmented network flows," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 11, no. 3, pp. 1–28, 2016. [Pg.104], [Pg.110]

[153] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM computer communication review*, vol. 35, no. 4, pp. 217–228, 2005. [Pg.104], [Pg.110]

[154] Z. Shi, J. Li, C. Wu, and J. Li, "DeepWindow: An efficient method for online network traffic anomaly detection," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2019, pp. 2403–2408. [Pg.105]

[155] F. S. d. Lima Filho, F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, "Smart detection: an online approach for DoS/DDoS attack detection using machine learning," *Security and Communication Networks*, vol. 2019, 2019. [Pg.105]

[156] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231. [Pg.111], [Pg.112]

[157] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122. [Pg.111]

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati 781039, India