



REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

**CYBER SECURITY SYSTEM FOR SDN
ENVIRONMENT TO DETECT THE DIFFERENT
ATTACKS BY UTILIZED HYBRID METHOD OF
DEEP LEARNING TECHNIQUES**

Ali Tariq Kalil AL-KHAYYAT

Doctor of Philosophy

Supervisor

Prof. Dr. Osman Nuri UÇAN

İstanbul, 2025

**CYBER SECURITY SYSTEM FOR SDN ENVIRONMENT TO
DETECT THE DIFFERENT ATTACKS BY UTILIZED HYBRID
METHOD OF DEEP LEARNING TECHNIQUES**

Ali Tariq Kalil AL-KHAYYAT

Electrical and Computer Engineering

Doctor of Philosophy

ALTINBAŞ UNIVERSITY

2025

The dissertation titled CYBER SECURITY SYSTEM FOR SDN ENVIRONMENT TO DETECT THE DIFFERENT ATTACKS BY UTILIZED HYBRID METHOD OF DEEP LEARNING TECHNIQUES prepared by ALI TARIQ KALIL AL-KHAYYAT and submitted on 10/06/2025 has been **accepted unanimously** for the degree of PhD in Electrical and Computer Engineering

Prof. Dr. Osman Nuri UÇAN

Supervisor

Thesis Defense Committee Members:

Prof. Dr. Osman Nuri UÇAN

Department of Electrical-
Electronic Engineering,

Altınbaş University

Assoc.Prof. Dr. Sefer KURNAZ

Department of Computer
Engineering,

Altınbaş University

Assoc. Prof. Dr. Oğuz ATA

Department of Software
Engineering,

Altınbaş University

Asst. Prof. Dr. Serdar KARGIN

Department of Biomedical
Engineering,

İstanbul Arel University

Prof. Dr. Adel Deniz DURU

Department of Sports and
Health Sciences

Marmara University

I hereby declare that this dissertation meets all format and submission requirements for a PhD thesis.

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Ali Tariq Kalil AL-KHAYYAT

Signature



ABSTRACT

CYBER SECURITY SYSTEM FOR SDN ENVIRONMENT TO DETECT THE DIFFERENT ATTACKS BY UTILIZED HYBRID METHOD OF DEEP LEARNING TECHNIQUES

AL-KHAYYAT, Ali Tariq Kalil

Ph.D., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof. Dr. Osman Nuri UÇAN

Date: 06/2025

Pages: 159

As contemporary networks come under attack with sophisticated and mass-scale cyber-threats, traditional intrusion detection systems (IDS) are unable to cope with changing threats. Such limitations are the focus of this research, and two new deep learning-based frameworks are introduced, which are best suited for fast and effective intrusion detection in Software-Defined Network (SDN) environments: (1) the Adversarial Learning-based Multi-Branched Hybrid Architecture and (2) the Multi-Branched Hybrid Perceptron Network (MBHPN). Together, these models offer an intelligent and scalable IDS platform that can detect advanced threats, in particular Distributed Denial of Service (DDoS) attacks, through adaptive learning, real-time integration, and context analysis. The first architecture, as explained in Chapter 3, combines Convolutional Neural Networks (CNN), Capsule Networks, and Long Short-Term Memory (LSTM) layers in a multi-branch architecture. CNNs encode spatial dependencies, Capsule Networks preserve hierarchical relations among features, and LSTMs describe sequential behaviors of traffic flows. The framework also incorporates Dynamic Adversarial Learning, where adversarial samples are produced to mimic attacks, thus enhancing model robustness. Each branch produces a unique feature representation, which is combined using attention-weighted methods to create an integrated, discriminative feature space tailored for intrusion detection. In Chapter 4, this architecture is enriched by the addition of MBHPN, which is directly optimized for DDoS attack

detection. This network combines three deep learning branches: MLP, DenseNet-like, and ResNet-like units. It adds Dynamic Feature Adaptation (DFA) to down-regulate noisy features and up-regulate pertinent traffic signatures. Multi-instance Learning (MIL) to process aggregated traffic flows instead of standalone instances, and thus greatly enhances contextual perception. These improvements make MBHPN robust against class imbalance, evasion attacks, and changing attack patterns prevalent in actual networks. The models are trained and tested on three intrusion detection benchmark datasets: UNSW-NB15, CICIDS2017, and CSE-CIC-IDS2018, which contain varied traffic patterns and attack conditions. Proven through extensive experimentation, the models outperform the current state of affairs. On the UNSW-NB15 dataset, MBHPN reports 99.31% accuracy, 98.02% precision, 98.87% recall, and an F1-score of 98.44% with a false positive ratio (FPR) decreased to 0.61% from the original 1.12% FPR of the base model. The time taken for inference reduced from 7.8 ms to 5.3 ms, which reflects a 32% boost in speed from SDNetc integration.

Keywords: Software-Defined Network, Convolutional Neural Networks, Cyber Security, False Positive Ratio, Dynamic Adversarial Learning.

TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT	v
LIST OF TABLES	xi
LIST OF FIGURES	xii
ABBREVIATIONS.....	xv
1. INTRODUCTION.....	1
1.1 BACKGROUND AND OVERVIEW.....	1
1.2 FOUNDATIONS OF IDS, DL TECHNIQUES, AND SDNETC	3
1.2.1 The Role of Deep Learning in IDS	3
1.2.2 Emergence of Software-Defined Networking (SDN).....	5
1.2.3 Bridging Deep Learning and SDN for Next-Gen IDS.....	6
1.3 IMPORTANCE OF INTRUSION DETECTION IN CYBERSECURITY	7
1.4 CHALLENGES IN INTRUSION DETECTION CLASSIFICATION.....	9
1.5 MOTIVATION FOR THE RESEARCH	11
1.6 OBJECTIVES OF THE THESIS	11
1.7 CONTRIBUTIONS OF THE THESIS	13
1.8 ORGANIZATION OF THE THESIS.....	14
2. LITERATURE REVIEW	15
2.1 OVERVIEW AND CATEGORIZATION OF IDS TECHNIQUES.....	15
2.1.1 Signature-Based IDS.....	16
2.1.2 Anomaly-Based IDS	18
2.1.3 Hybrid Intrusion Detection Systems (IDS).....	22
2.2 MACHINE LEARNING AND DEEP LEARNING IN IDS.....	24
2.2.1 Traditional Machine Learning Approaches	24
2.2.2 Deep Learning for Intrusion Detection	27
2.3 ADVERSARIAL ATTACKS IN INTRUSION DETECTION.....	31
2.3.1 Types of Adversarial Attacks on IDS	32

2.3.2 Countermeasures Against Adversarial Attacks	34
2.4 DDOS ATTACK DETECTION AND MITIGATION	37
2.4.1 Types of DDoS Attacks	38
2.4.2 Machine Learning-Based DDoS Detection	39
2.5. IDS IN SDN AND SDNET ENVIRONMENTS.....	41
2.6 SUMMARY AND RESEARCH GAPS	43
3. PROPOSED METHODOLOGY ONE	46
3.1 OVERVIEW OF THE PROPOSED MODEL.....	46
3.2 DATASETS FOR INTRUSION DETECTION	48
3.2.1 NSL-KDD Dataset	48
3.2.2 CICIDS 2017 Dataset	49
3.2.3 CIC-DDoS2019 Dataset.....	49
3.3 DATA PRE-PROCESSING TECHNIQUES	50
3.3.1 Scaling.....	51
3.3.2 Encoding	53
3.3.3 Data Splitting	54
3.4 MULTI-SCALE SPARSE CNN FOR SPATIAL FEATURE EXTRACTION.....	56
3.5 GRID LSTM FOR TEMPORAL DEPENDENCY LEARNING.....	57
3.6 STACKED CAPSULE NETWORKS FOR HIERARCHICAL FEATURE LEARNING	58
3.7 DECISION-LEVEL FEATURE FUSION STRATEGY	59
3.8 ADVERSARIAL LEARNING WITH THRESHOLD-BASED ATTACK DETECTION.....	60
3.9 SUMMARY	62
4. PROPOSED METHODOLOGY TWO	65
4.1 INTRODUCTION TO DDOS ATTACK CLASSIFICATION	65
4.2 DATASETS FOR INTRUSION DETECTION	71
4.2.1 UNSW_NB15 Dataset	71
4.2.2 CICIDS2017 Dataset	73

4.4.3 CIC-DDoS2019 Dataset.....	73
4.3 DATA PRE-PROCESSING TECHNIQUES	75
4.4 DYNAMIC FEATURE ADAPTATION MECHANISM	77
4.5 MULTI-BRANCHED HYBRID PERCEPTRON NETWORK.....	81
4.6 FEATURE FUSION BY DYNAMIC ATTENTION-WEIGHTED FEATURE FUSION	82
4.7 EXPERIMENTAL SETUP.....	86
4.8 INTEGRATION OF SDNETC WITH MHHPN-BASED DETECTION FRAMEWORK	86
4.8.1 Centralized Control and Programmability in SDNetc for MHHPN Deployment	87
4.8.2 Real-Time Traffic Redirection for Dynamic Analysis	87
4.8.3 Enhanced Scalability and Real-Time Responsiveness	88
4.8.4 Pattern of Attacks.....	88
4.9 SUMMARY	89
5. RESULTS AND DISCUSSION.....	91
5.1 INTRODUCTION	91
5.2 PERFORMANCE EVALUATION METRICS IN IDS.....	91
5.2.1 Accuracy, Precision, Recall, F1-Score, Confusion Matrix.....	91
5.2.2 AUC-ROC and Matthews Correlation Coefficient (MCC)	93
5.2.3 Detection Rate and False Positive Rate	93
5.3 PERFORMANCE COMPARISON OF PROPOSED MODELS.....	94
5.4 ABLATION STUDY OF INDIVIDUAL COMPONENTS.....	116
5.4.1 Effect of Sparse CNN in Feature Extraction.....	116
5.4.2 Effect of Grid LSTM on Temporal Learning	117
5.4.3 Role of Capsule Networks in Hierarchical Feature Learning.....	117
5.4.4 Impact of Multi-Instance Learning on Detection Accuracy	117
5.4.5 Applicability of the Proposed Architecture to SDN Environments.....	118
5.5 MODEL COMPLEXITY AND COMPUTATIONAL EFFICIENCY	118
5.5.1 Real-Time DDoS Detection with SDNetc-Integrated MHHPN Framework.....	122

5.6 DISCUSSION ON GENERALIZATION CAPABILITY	124
5.6.1 Discussion on Generalization Capability	124
6. CONCLUSION AND FUTURE WORK.....	125
6.1 CONCLUSION.....	125
6.2 FUTURE WORK.....	126
REFERENCES	127



LIST OF TABLES

	<u>Pages</u>
Table 2.1: Representative IDS Frameworks for SDN and SDNet Environments	42
Table 3.1: Description of the Dataset Utilized in the Present Study [159]	53
Table 4.1: Data Partitioning of Proposed Model on Datasets	74
Table 5.1: Optimal Value of Several Hyper-parameters Associated with Proposed Approach (on NSL-KDD Dataset)	95
Table 5.2: Comparative Assessment of the Proposed Approach with SOTA Classification Models (in terms of Precision, Recall and F1-measure)	96
Table 5.3: Performance Analysis of the Proposed Approach on CIC-IDS and UNSW-NB15 datasets	98
Table 5.4: Parametric Architecture of the MHHPN Network.....	101
Table 5.5: Hyper-parameter Configuration and Optimization for the MHHPN Network	105
Table 5.6: Comparative Performance Between Baseline Models and Proposed Model on Dataset 1	106
Table 5.7: Comparative Performance Between Baseline Models and Proposed Model on Dataset 2	108
Table 5.8: Comparative Performance Between Baseline Models and Proposed Model on Dataset 3	112
Table 5.9: Experimental Results Assessment with Varying Hyper-Parameters	119
Table 5.10: Comparative Evaluation of MHHPN With and Without Sdnetc Integration (UNSW-NB15 Dataset).....	123

LIST OF FIGURES

	<u>Pages</u>
Figure 1.1: Architecture of Basic IDS [25].	2
Figure 1.2: SDN-Based Intrusion Detection System Integrated with Deep Learning for Real-Time Threat Detection and Response [40]	5
Figure 1.3: Threat Landscape and Challenges in Intrusion Detection [24]	10
Figure 2.1: Organization of A Generalized Intrusion Detection System [26]	16
Figure 2.2: Feature Selection Process [67]	21
Figure 2.3: Architecture of Hybrid IDS [56]	22
Figure 2.4: Model Training and Evaluation [72]	26
Figure 2.5: Overview of NIDS Model Using CICIDS-2017 Dataset [75]	27
Figure 2.6: Flow of the CNN-LSTM [79]	28
Figure 2.7: Structure of LSTM [78]	29
Figure 2.8: Adversarial Attack Scenario with Ongoing Training [87]	33
Figure 2.9: Depiction of the Model Inversion Attack Process [93]	34
Figure 3.1: Working Methodology of the Proposed Multi Branched Architecture	47
Figure 3.2: Basic CNN Architecture	48
Figure 3.3: LSTM Cell Architecture	58
Figure 3.4: Representation of Features Fusion and Adversarial Learning	59
Figure 3.5: Workflow of the Proposed Model	61
Figure 3.6: Illustrates the Pseudo-code for Proposed Model	62
Figure 4.1: Working with Botnets to Execute A DDoS Attack	67

Figure 4.2: Pipeline Architecture of The Proposed Deep Learning-Based IDS Model	70
Figure 4.3: Shows the Steps of The Pseudocode for Dynamic Feature Adaptation.....	79
Figure 4.4: Illustrates Pseudocode for Dynamic Feature Adaptation.....	80
Figure 4.5: The Architecture of The Proposed Hybrid Complex Multilayer Perceptron (MHHPN) Network.....	81
Figure 4.6: Illustrates Pseudocode for Feature Fusion Process by Dynamic Attention-Weighted Mechanism	84
Figure 4.7: Pseudocode for Feature Fusion Process by Dynamic Attention-Weighted Mechanism	85
Figure 5.1: Layout of The Confusion Matrix for The Bi-Classification Problem.....	92
Figure 5.2: State-of-the-art Classification Models (on NSL-KDD Dataset).....	96
Figure 5.3: Assessment of Model Training Results in Terms of (a) Training Accuracy (b) Loss and (c) ROC (NSL-KDD Dataset)	97
Figure 5.4: Performance Analysis of The Proposed Approach On CIC-IDS And UNSW-NB15 Datasets	98
Figure 5.5: Accuracy Comparison with Benchmark Models on UNSW-NB15 Dataset [190]	99
Figure 5.6: Accuracy Comparison with Benchmark Models on UNSW-NB15 Dataset [190]	100
Figure 5.7: Comparative Performance Between Baseline Models and Proposed Model on Dataset 1	106
Figure 5.8: Confusion Matrix for Dataset 1	107
Figure 5.9: The Performance Curve of The Proposed Methodology on the UNSW-NB15 Dataset	108

Figure 5.10: The ROC Curve Shows the Discriminability Power of the Proposed Model on Dataset 1	108
Figure 5.11: Comparative Performance Between Baseline Models and Proposed Model on Dataset 2	109
Figure 5.12: Confusion Matrix For CIC-IDS2017 Dataset	110
Figure 5.13: The ROC Curve Shows the Discriminability Power of The Proposed Model on Dataset 2	111
Figure 5.14: The Performance Curve of The Proposed Methodology on the CIC-IDS2017 Dataset	112
Figure 5.15: Comparative Performance Between Baseline Models and Proposed Model on Dataset 3	113
Figure 5.16: Confusion Matrix for Dataset 3	115
Figure 5.17: The ROC Curve Shows the Discriminability Power of The Proposed Model on Dataset 3	115
Figure 5.18: The Performance Curve of The Proposed Methodology on the CSE-CIC-IDS2018 Dataset.....	116
Figure 5.19: Experimental Results Assessment with Varying Hyper-Parameters	119
Figure 5.20: Time Complexity (In Seconds) Comparison on All Three Datasets Among All the Methods	121
Figure 5.21: Comparative Evaluation of MHHPN With and Without SDNetc Integration (UNSW-NB15 Dataset).....	123

ABBREVIATIONS

APT	:	Advanced Persistent Threat
BN	:	Batch Normalization
CNN	:	Convolutional Neural Network
CSRF	:	Cross Site Request Forgery
DDoS	:	Distributed Denial of Service
DNN	:	Deep Neural Network
FCN	:	Fully Connected Network
FPR	:	False Positive Rate
GRU	:	Gated Recurrent Unit
IAM	:	Identity and Access Management
IDS	:	Intrusion Detection System
LSTM	:	Long Short-Term Memory
MITM	:	Common Vulnerabilities and Exposures
PSNR	:	Man-In-The-Middle
RLU	:	Rectified Linear Unit
SDN	:	Software Defined Networking
SIEM	:	Security Information and Event Management
SOC	:	Security Operations Center
TLS	:	Transport Layer Security
TPR	:	True Positive Rate
VPN	:	Virtual Private Network

1. INTRODUCTION

The advent of the explosive proliferation of internet-of-things devices and the adoption of cloud services, IoT, and distributed architectures has exposed networks to unprecedented security threats. Among all the techniques for securing networked systems, Intrusion Detection Systems (IDS) are at the leading edge in detecting unauthorized access, malicious behavior, and anomalous traffic patterns [1], [2]. With the increasing sophistication of cyber-attacks, the traditional IDS approaches fall behind the pace [3], [4], and hence the trend toward deep-learning-based intelligent and adaptive detection systems [5], [6] and the next-generation networking architectures like Software Defined Networking (SDN) and SDNets (SDN enabled networks with built-in edge/cloud intelligences and cooperative architectures) [7], [9] This chapter sets out the background, context, problem, and motivation for this research, establishing the scope and significance of the thesis.

1.1 BACKGROUND AND OVERVIEW

DDoS (Distributed Denial of Service) attacks endanger network integrity and system performance at their very essence by methodically draining target resources with massive volumes of traffic. [18]. DDoS attacks intend to drain the resources of a target system such as servers, network bandwidth, or applications by overwhelming them with vast volumes of false traffic from numerous sources. [19]. The very high frequencies of the DDoS attacks pose a colossal challenge to detecting and preventing them, and advanced techniques have to be utilized to enable efficient classification and defense [20]. Excessive data to be produced is one of the key challenges to DDoS attack detection. DDoS-hit networks face gigantic traffic, and hence it is difficult for the conventional security management systems to handle and analyze the incoming data effectively [21]. The volume of data is so huge that it threatens real-time monitoring as well as adds additional burden on computational resources, which calls for more scalable and efficient solutions. To boot, the evolving character of the DDoS attacks dynamically provides an additional complexity level. Attacker also characteristically alters its tactics in a desperate attempt at evasion. They employ varying techniques such as traffic obfuscation and spoofing [22]. As a result, there is a need for continuous updates and refinements to the detection modules so that they can easily adapt to these changing patterns. This dynamic aspect makes conventional identification

methods obsolete, highlighting the significance of adaptive and flexible classification systems. The second critical challenge is associated with the class imbalance problem. In the typical network traffic data scenarios, normal traffic samples appear comparatively a larger number of times than the attack traffic samples. This creates a unbalanced distribution of the overall data and can negatively impact the performance of the classification models [25]. The learning models trained on these skewed distribution fails to generalize well on the given dataset and tends to favor the class with a larger number of samples. This leads to poor detection rates for less frequent attack types. Addressing this crucial issue is significant for building the models capable of accurately identifying and classifying various DDoS attacks. Figure 1.1 illustrates the Basic architecture of Intrusion Detection System.

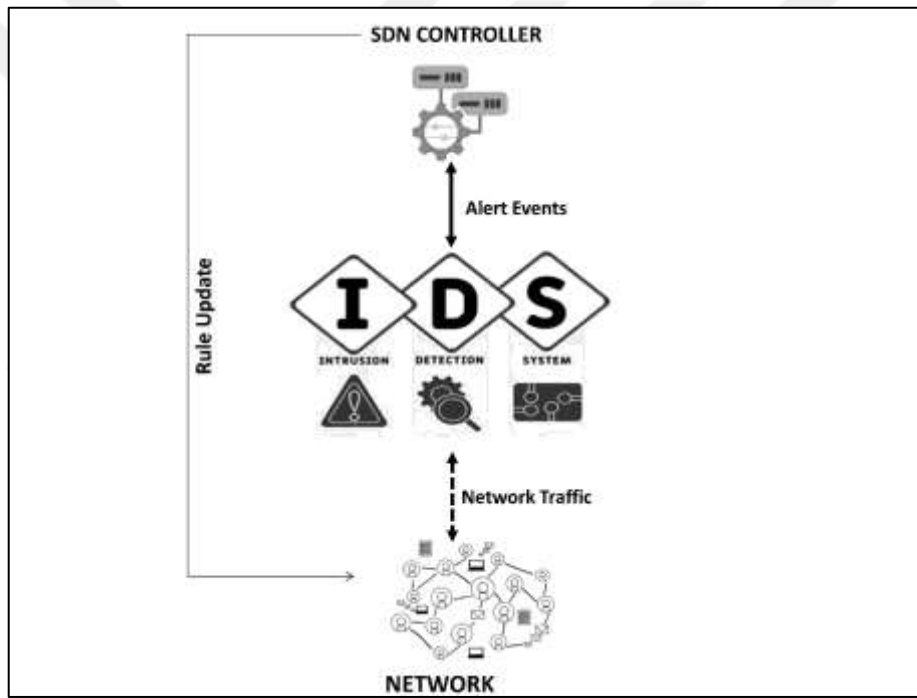


Figure 1.1: Architecture of Basic IDS [25].

Moreover, adversarial attacks add another layer of complexity to identify the attacks efficiently [26]. Malicious intruders may implement several different kinds of techniques to deceive or deceive the detection module, thus exploiting vulnerabilities in the model to evade detection. These adversarial tactics can result in false negatives or reduced detection accuracy. Hence, it is essential to develop robust models capable of handling such manipulations [27], [28]. The capability to detect and counteract adversarial attacks is essential for maintaining the reliability and effectiveness of classification systems. Real-time

processing is another critical requirement for DDoS detection. With the growing network traffic and potentially devastating impact of DDoS attacks, it is critical to timely detect and respond against these attacks. It will help in mitigating the damage and ensuring network stability. Hence, the detection models must be capable of processing incoming traffic swiftly and accurately for prompt identification and initiation of defensive measures. This necessitates high performance models that can handle the large and complex network-traffic in real time [29].

1.2 FOUNDATIONS OF IDS, DL TECHNIQUES, AND SDNETC

IDS are essential components in the cyber security infrastructure, designed to detect misuse, illegal access, or compromise of computer systems and networks. Their core function is to observe network traffic or the actions happening in system and raise alerts when potential malicious activities are detected. Intrusion detection systems help administrations categorize potential security breaks in real-time and are often deployed as part of a multi-layered defense strategy to strengthen system resilience. Anomaly-based and Signature-based systems are types of IDSs [30].

Signature-based IDS detect known threats through network traffic and a predefined database of attack patterns or “signatures” [31]. They are efficient and highly accurate for recognizing previously encountered threats. However, their primary limitation is their inability to detect novel or zero-day attacks, as these threats lack existing signatures in the database [132]. In contrast, anomaly-based IDS function by creating a starting point of usual network performance and identifying deviances from starting point as likely intrusions [32], [34]. This category of IDS is better suited to find hidden dangers adaptable to evolving cyber threat landscapes. Anomaly-based systems are prone to greater FPR (false positive rates) and require more complex analysis techniques to distinguish between benign anomalies and malicious behaviors.

1.2.1 The Role of Deep Learning in IDS

As network data sizes and complexity increase, traditional intrusion detection techniques yield high false positives and low accuracy. This has witnessed the rise of mounting interest in the use of deep learning (DL) techniques that have been tremendously successful in other domains such as computer vision, natural language processing, and time-series based

forecasting. Deep models can learn hierarchical representations of raw data and non-linear relationships on their own and are therefore most appropriate to be applied in intrusion detection systems [35], [36], [37]. Among all other deep architectures used in IDS, CNNs are most appropriate to manage network traffic spatial relationships, namely payload inspection and packet authentication [38]. CNNs learn local patterns from data that may have spatial patterns indicating intrusion. RNNs and their variants such as LSTM networks are best suited to represent sequential data and hence best used in temporal pattern analysis of network flow traffic [39]. RNNs can learn time-dependent patterns, which are highly beneficial to identify time-evolving attack behavior such as DDoS or brute-force attacks. Auto-encoders are also widely applied for unsupervised anomaly detection, where the network learns to reconstruct normal behavior and mark inputs with high reconstruction errors as anomalies [40]. Transformer models, which have recently set new state-of-the-art benchmarks in NLP, are now being explored for IDS because they can model long-range dependencies in network sessions using attention mechanisms. The use of the aforementioned deep learning models in IDS allows systems to learn high-level abstraction from high-dimensional input data, detect unknown attacks, and reduce false positives and false negatives significantly [41]. Figure 1.2 depicts an SDN (Software Defined Networking) based IDS architecture with integration of deep learning models for enhanced network security.

At the core of the design is the network layer, the actual data itself moving from one device to another. The network traffic is under constant surveillance by the IDS system. Traffic flows first through the monitor module, a real-time observation layer to scan for abnormality or suspicious behavior. It forwards the data to the IDS engine, which runs on advanced deep learning algorithms such as MLP (Multi-Layer Perceptron), DenseNet, and ResNet. These algorithms are fine-tuned for accurate detection of subtle malicious patterns. The IDS engine is supported by a response module that determines and applies suitable countermeasures depending on the nature and level of the identified threat. After detecting the intrusion or anomaly, an alert event is propagated to the controller of the SDN. The controller of the well-behaved network accepts and processes the alert and adds new flow rules into the network devices to react against attacks such as blocking malicious nodes or traffic directing. This creates a feedback loop where the SDN controller is constantly refining network policy based on input from the IDS in real time, allowing it to have a dynamically adaptive,

responsive security profile. Incorporating deep learning as a component in this system, overall capability for detection of sophisticated cyber-attacks and further elimination of false positives greatly enhances, making it extremely suitable to be implemented in programmable networks of today.

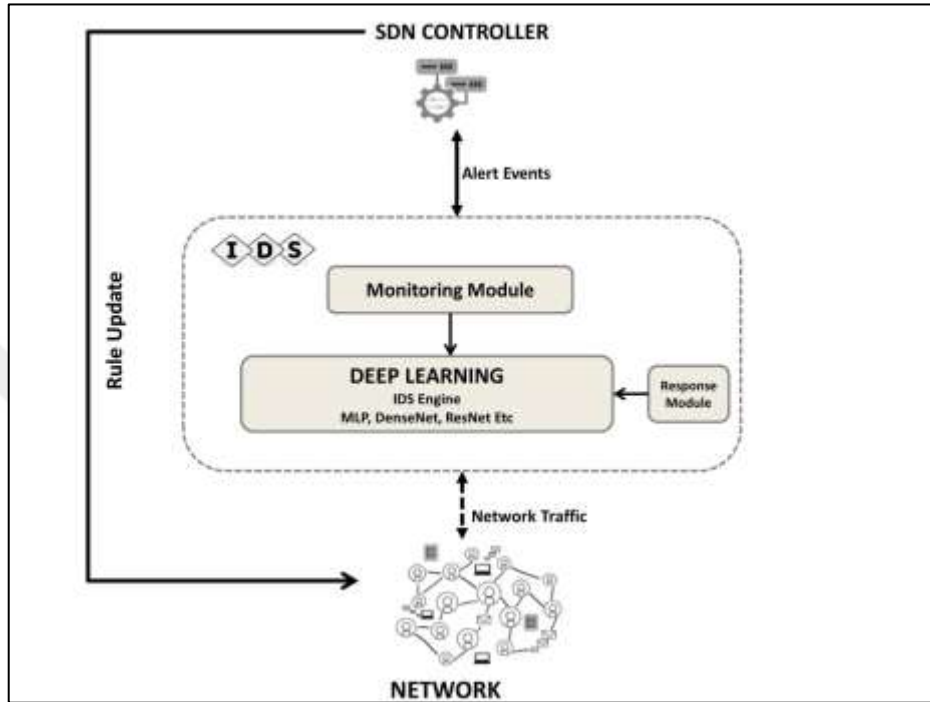


Figure 1.2: SDN-Based Intrusion Detection System Integrated with Deep Learning for Real-Time Threat Detection and Response [40]

1.2.2 Emergence of Software-Defined Networking (SDN)

Traditional network architecture with tightly coupled data plane and control plane possesses poor programmability, scalability, and flexibility. Control plane separation from data plane provides programmable and centralized network control [42]. In SDN, the entire network is controlled by a centralized controller that coordinates with autonomous forwarding devices (e.g., switches and routers) through standardized protocols such as OpenFlow. This architecture gives end-to-end visibility of the network, is easier to manage, and provides real-time feedback in the guise of routing and policy rules based on real traffic flows. SDN has created new possibilities for how to embed intelligent security function within the network itself. In giving visibility and real-time access to traffic flow, SDN allows threats to be monitored and responded to more actively. Moreover, SDN's centralized nature is best

positioned to support deep learning-based IDS solutions because it enables data collection, centralized processing, and dynamic reconfiguration of security policy [43].

1.2.3 Bridging Deep Learning and SDN for Next-Gen IDS

SDNet, or SDN + deep learning, is the integration of these two ground-breaking paradigms to create highly adaptive and intelligent network security systems. SDNet designs leverage the flexibility and programmability of SDN and the powerful pattern recognition capability of deep learning algorithms to enhance intrusion detection ability beyond conventional means. The primary advantage of SDNet is that it can observe, learn, and respond within a closed loop. The SDN controller collects rich information on the flow level and sends them to a deep learning module to perform threat detection. After the detection of a threat, the controller is able to perform countermeasures dynamically, i.e., block malicious flows or send them for further inspection, in real-time [44]. The components of a SDNet system are:

- i. Data Plane: Made up of SDN switches or routers, which forward packets based on flow rules guided by the controller.
- ii. Control Plane: The SDN controller, the network's heart, manages flow rules and administers security action.
- iii. Monitoring Module: Collects traffic statistics and flow data to inspect.
- iv. Deep Learning IDS Engine: Employs models like CNNs, LSTMs, or Transformer networks for anomaly detection and traffic marking as benign or malicious.
- v. Response Module: Dynamically changes flow rules in the controller depending on the output of the IDS engine to neutralize threats.

This tight feedback loop enables SDNet to offer real-time scalability, adaptability, and application-specific security enforcement, making it a highly promising architecture for next-generation network defense systems. One of the best applications of SDNet is for detecting DDoS attacks, in which the deep learning networks based on LSTMs and CNNs can identify anomalously huge bursts of traffic, and the SDN controller can reroute or block the malicious traffic automatically [45]. Another benefit of SDNet is that it isolates the traffic at a fine level so that highly contextual and specific responses to threats are possible.

SDNet infrastructure supports distributed infrastructures' cooperative security, or multi-cloud and hybrid networks. The centralized control of SDN and deep learning's capability to learn from experience allow SDNet infrastructure to identify coordinated multi-segment attacks on a network.

There are some challenges in using SDNet. For example, injecting deep learning models into the SDN control plane can cause computational overhead, latency, and scalability issues. High-efficient deployment techniques like utilizing edge computation or model compression need to be implemented in an attempt to resolve these constraints. Deep learning-based IDS in SDNet also considerably depends on clean labeled datasets that may not necessarily be readily accessible in real applications. Data pre-processing, feature extraction, as well as semi-supervised machine learning methods still need to be explored to incorporate models with enough robustness and generalization abilities. Last but not least, the interpretability and explainability of deep learning models in critical infrastructures like SDNet is a foremost issue. Employing Explainable AI (XAI) techniques will be an essential aspect of building trust and allowing regulatory compliance. The deployment of DL(Deep Learning) together with IDS has played a substantial role in empowering existing security systems to detect complex attacks. With the advent of SDN, network designs have become programmable and dynamic, and a flawless platform awaits the deployment of smart IDS. The advent of SDNet represents a paradigm shift towards real-time, adaptive, and autonomous models of security. With the power of both SDN and deep learning combined synergistically, SDNet offers an effective, scalable, and context-aware solution to the security of modern communication networks [46], [50].

1.3 IMPORTANCE OF INTRUSION DETECTION IN CYBERSECURITY

IDS are hardware or software products that detect unauthorized access, malicious behavior, or policy violations by continuously monitoring system or network activity. IDS's main function is to provide an early indication of potential security compromise so that the organization has ample time to respond quickly and minimize damage. Unlike preventive programs, Intrusion Detection System are more concerned with detection and notification. Real-time attack detection and notification are one of the primary applications of IDS.

Through constant monitoring of network traffic and user activity, Intrusion Detection System can detect anomalous patterns of activity, such as suspicious login attempts, privilege escalation, or large data transfer, and trigger real-time alarms [1], [13]. The alarms can trigger automatic or manual response actions to quarantine the threat. In most cases, early detection bars the attacker from proceeding further into the system, thereby limiting damage. Second, IDS enables the minimization of the incident response time by making use of intelligence and the threat context that is already known [4]. Through the quick determination of the source, the attack type, and the impact of an attack, remediation activities can be properly prepared and set in motion. The response time should be fast in a manner that minimizes breach impact, especially when downtime or data loss would be catastrophic. Thirdly, IDS complements other security devices such as antivirus software, firewalls and SIEM (Security Information and Event Management) systems [18]. Whereas firewalls block traffic based on pre-configured rules and antivirus software detects known malware, IDS goes a step further by inspecting traffic behavior and detecting known and unknown threats, including insider threats and zero-day attacks. Such a multi-layered defense system significantly enhances an organization's overall security posture. IDS provide valuable information on system weaknesses and likely security vulnerabilities [19].

By recording events of known anomalies and intrusion attempts, IDS enables organizations to identify vulnerable areas of their networks and implement tactical adjustments. The logs also assist in forensic analysis and compliance reporting, which are required for auditing and post-incident reporting. In high-criticality industries like healthcare, finance, energy, and government, the need for IDS is even more critical. These sectors handle humongous volumes of sensitive and personally identifiable information (PII), and any breach can have disastrous consequences. IDS help to provide data integrity, data availability and data confidentiality, adherence to the regulation as well as perfect provision of service [20]. IDS play a significant function in safeguarding digital assets because threats within the cyberspace become wiser and ruthless. By allowing the early detection of threats, minimizing response time, aiding other security controls, and emphasizing vulnerabilities, IDS assists organizations in creating effective and responsive cyber security infrastructures that can withstand today's threats.

1.4 CHALLENGES IN INTRUSION DETECTION CLASSIFICATION

Despite the fact that Intrusion Detection Systems have reached great heights with the implementation of artificial intelligence and deep learning, there are still some issues that impede their performance in actual deployment. These issues not only impact detection accuracy but also undermine the scalability and flexibility of IDS in heterogeneous and dynamic environments.

- i. **High False Positive Rate (FPR):** The most traditional issue of anomaly-based IDS is high FPR. Such IDS mark intrusions by detecting deviation from what is "normal" behavior. It is intrinsically difficult to detect normal behavior in complex and heterogeneous network configurations. Therefore, much legitimate behavior is mistakenly marked as malicious [11], [21]. This leads to multiplication of false alarms, and these may delay security analysts, as well as result in alert fatigue. Under severe conditions, true threats might be missed due to persistent false alarms.
- ii. **Imbalanced Datasets:** The second critical issue is the presence of imbalance in training datasets, wherein examples of malicious traffic are highly dominated by examples of normal traffic [22]. The majority of supervised machine learning algorithms are inappropriate for imbalanced data as they get biased towards the majority class. IDS may get biased to mark traffic as normal and hence miss out on unusual but notable attack patterns. This issue is particularly severe in the training data of the deep learning model, where even the best architectures are unable to learn good representations for out-of-distribution attack classes.
- iii. **Dynamic Threat Environment:** Cyber-attacks are on the rise as attackers continually introduce new techniques, tactics, and procedures (TTPs). Dynamic threat environment is an indication that static models of IDS become extremely obsolete unless retrained a few times with new data [23]. Zero-day attack or variant detection of the previously known attacks remains a challenging process. Not only should the IDS detect known threat, but it should also generalize sufficiently to detect new variants that are characteristics of new attacks. **Latency and Scalability Issues:** In real-time environments, IDS must process massive volumes of traffic with minimal latency. Deep learning models especially those with complex architectures like recurrent or transformer-based networks can introduce processing delays that hinder real-time intrusion detection [9], [16].

Moreover, as the volume and speed of network traffic increase, maintaining both high accuracy and low latency becomes more challenging. Scalability is also an issue when deploying IDS across large, distributed networks, requiring efficient data processing pipelines and resource optimization.

- iv. Lack of Generalizability: A frequent problem with IDS models is the lack of generalizability across different datasets or network environments [14], [24]. A model trained on a specific dataset may perform well in that context but fail to detect threats when deployed in a different environment due to variations in network behavior, traffic patterns, or attack signatures. This indicates a need for domain adaptation techniques or models capable of learning invariant features across diverse conditions.

Figure 1.3 presents a conceptual overview of the cyber security landscape, highlighting the major types of cyber threats, the challenges they introduce, and the potential solutions offered by deep learning-based approaches. On the left side, common forms of cyber threats are depicted, including Distributed Denial of Service attacks, brute force attacks, infiltration attempts, and data exfiltration. These threats pose significant risks to modern network environments.

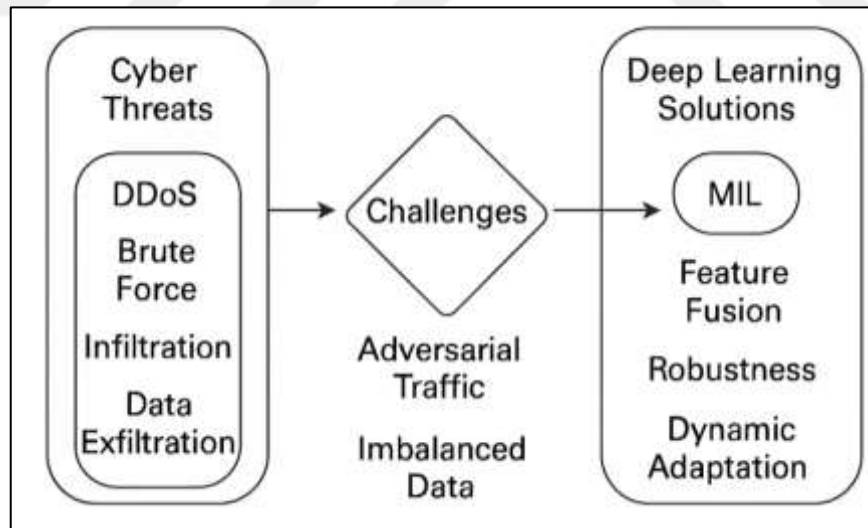


Figure 1.3: Threat Landscape and Challenges in Intrusion Detection [24]

These threats give rise to several challenges in designing effective intrusion detection systems. Two key challenges highlighted in the figure are adversarial traffic, where attackers manipulate inputs to evade detection systems, and imbalanced data, which refers to the disproportionate representation of attack vs. normal traffic in training datasets. These issues

complicate the learning process and reduce the reliability of traditional machine learning approaches. Application of deep learning solutions is shown to address the challenges by utilizing Multi-Instance Learning (MIL). MIL is capable of learning from groups of events instead individual samples, which helps in better generalization, especially in ambiguous or partially labeled data scenarios. Additional strategies include feature fusion (combining diverse data representations for richer context), robustness (ensuring the model performs well even with noisy or adversarial data), and dynamic adaptation. All these approaches, combined, offer an intelligent and more responsive security measure against dominant cyber-attacks.

1.5 MOTIVATION FOR THE RESEARCH

With today's highly networked digital world, greater dependency on Software Defined Networking (SDN) to deliver agile, programmable, and scalable infrastructure means new types of threat are not easily mitigated by legacy cyber defense strategies [38]. Of those, Distributed Denial of Service (DDoS) attacks represent one of the strongest types of threat due to their ability to cause entire networks to become unusable through bandwidth saturation, server saturation, or poisoning of control planes. Existing intrusion detection systems will not be able to dynamically adjust in real-time to new types of attacks and conventional rule-based or shallow learning methods generate uncontrolled high false positives without robustness against new threats. Motivated by these limitations, this research proposes a novel Multi-Branched Hybrid Perceptron Network (MHHPN) that integrates several deep learning architectures MLP, DenseNet, and ResNet with dynamic feature adjustment and multi-instance learning to provide a robust, adaptive detection framework for SDN networks [41]. The addition of MHHPN to the SDN system offers intelligent re-routing of traffic, real-time threat identification, and dynamic policy updates, offering a next-generation programmable network-specific integrated solution.

1.6 OBJECTIVES OF THE THESIS

Appearance of more advanced network infrastructures and emergence of programmable models like Software Defined Networking (SDN) have made the traditional cyber security practices obsolete. The thesis tries to design an abstract, smart, and learning intrusion

detection system taking benefit of the capability of deep learning in avoiding novel cyber-attacks upon such infrastructures. The primary objectives of the thesis are:

- i. To explore and develop a hybrid deep learning model that effectively learns different patterns of network traffic in order to effectively detect threats. This objective is towards developing a new hybrid model by combining the strengths of different deep learning models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and feed-forward layers in an attempt to explore spatial and temporal patterns of network traffic. The intention is to create a strong model to distinguish between malicious and benign activity, including advanced and evolving patterns of attack.
- ii. Designing an adaptive detection system that changes configuration through adapting attack behavior and feature distributions in real time. Because cyber-attacks are said to change their signatures and modalities, a static model detection becomes outdated in a few seconds. This objective imposes the necessity for a dynamically reconfigurable system with the capacity to address dynamic traffic patterns, evolving to changing attack surfaces, and maintaining high detection rates whenever concept drift and adversarial inputs are involved.
- iii. To include learning mechanism that enhances the model to understand and process group instances of network traffic in a manner that facilitates context-aware detection. Distributed Denial of Service (DDoS) attacks, botnet behavior, and scan attack coordination will be anticipated on groups of group instances of traffic rather than packets or flows. This objective target building multi-instance learning and clustering algorithms for enabling the model to acquire contextual data from batches of akin traffic data in order to enhance sophisticated and covert attack detection.
- iv. To implement the suggested detection framework on a programmable network platform in order to facilitate centralized control and effective traffic handling. impressed with the power of centralized SDN controller and its awareness throughout the entire network, the described framework will run over SDN-compliant infrastructures. It will enable real-time detection of threats, effective provisioning of rules, and immediate response against detected threats. It will run on top of SDN controllers like OpenDaylight or Ryu, it will be scalable, and installable over live networks.
- v. Testing the suggested system with widely used benchmark data sets to verify that its scalability, usability, and generality are tested in various network environments. The

system for detection will be adequately tested on widely used public data sets like NSL-KDD, CICIDS2017, or UNSW-NB15. The data sets have many attacks and network traffic and are therefore most appropriate to test the system performance under varying real-time situations. Performance will be evaluated on the basis of metrics such as accuracy, recall, precision, F1-score, and false positive rate.

- vi. Comparison of the model's performance against other approaches and if it is useful in reducing false positives without sacrificing detection capacity. The goal is a comparison of the provided model against other intrusion detection systems ranging from basic machine learning models to the state-of-the-art deep learning models available at present. Comparison will also involve a thorough description of how the proposed model stands in terms of performance, computational complexity, and deployment feasibility. Emphasis will be placed on minimizing false positives, which is one of the key issues when it comes to managing intrusion detection, as well as improving the detection rate against different types of attacks.

1.7 CONTRIBUTIONS OF THE THESIS

This thesis is achieving some unprecedented advancement in SDN settings towards cyber security and smart intrusion detection. With the virtue of leveraging deep learning advancements to real-time SDN functions, this thesis is introducing a scale-out, high-level, versatile network security solution that can be supported by any infrastructure. This thesis proposes a hybrid deep learning-based IDS framework, tailored for SDN environments, capable of detecting a wide range of network attacks with high accuracy and adaptability. Following highlights, the contribution to the thesis.

- i. A novel hybrid deep learning architecture (MHHPN) combines MLP, DenseNet, and ResNet branches to exploit complementary feature learning capabilities.
- ii. A dynamic feature adaptation mechanism to refine feature importance using attention-based weighting, addressing feature redundancy and noise.
- iii. An effective use of Multi-Instance Learning (MIL) to model flow-level attack patterns from grouped data, enhancing contextual detection of complex attacks.
- iv. Development of a dynamic attention-weighted feature fusion strategy for merging multi-branch outputs, leading to improved generalization and robustness.

- v. Seamless integration of the MHHPN within an SDNetc framework, enabling real-time traffic redirection, dynamic threat response, and adaptive policy updates in programmable networks.
- vi. Empirical validation across three real-world intrusion detection datasets, achieving superior accuracy and robustness compared to existing approaches, demonstrating applicability to SDN-enabled infrastructures.

1.8 ORGANIZATION OF THE THESIS

This thesis is organized into six chapters, each building upon the previous to present a comprehensive study on the development of a cyber-security system for SDN environments using hybrid deep learning techniques.

Chapter 1 provides an overview of the research problem, motivation, objectives, contributions, and the scope of the study. Chapter 2 gives detailed survey of existing approaches in intrusion detection, with a focus on SDN-based systems and hybrid deep learning methods. It identifies research gaps and lays the foundation for the proposed methodology. Chapter 3 introduces the core hybrid model, detailing the design of the Multi-Branched Hybrid Perceptron Network (MHHPN), data selection, and the underlying methodology for DDoS detection. Chapter 4 enhances the architecture introduced in Chapter 3 with advanced techniques such as dynamic feature adaptation, multi-instance learning, and SDN integration. It also presents dataset preparation, pre-processing, and model optimization techniques. Chapter 5 presents a comprehensive analysis and evaluation of the proposed system using benchmark datasets. It also includes comparisons with existing models, ablation studies, and generalization performance discussions. Chapter 6 contains the primary findings of the work, outlines contributions, and maps out avenues for further extension, particularly real-time application and full attack coverage.

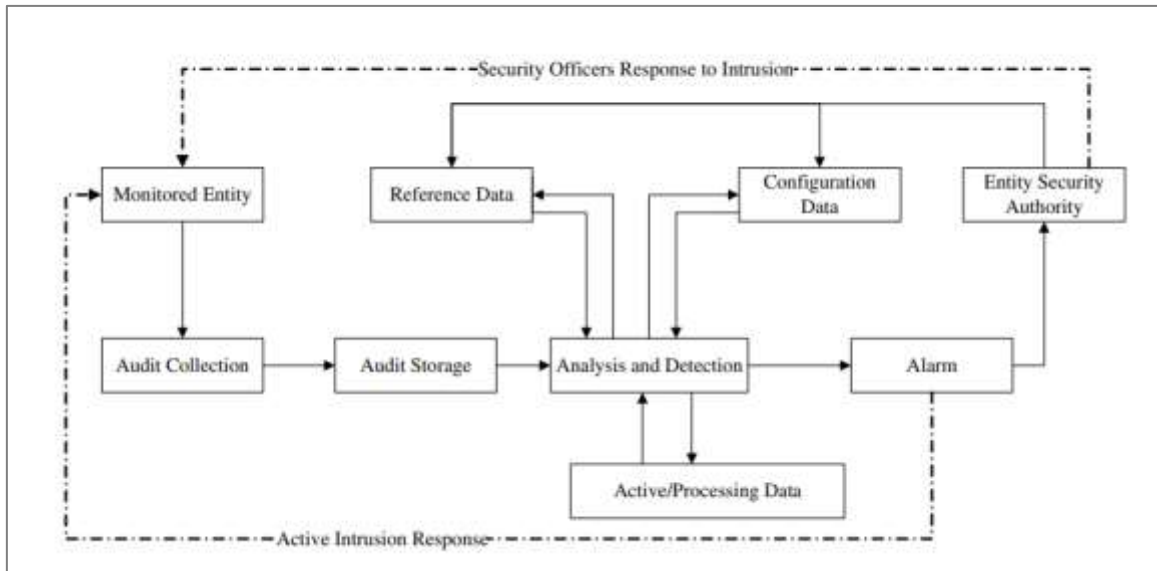
2. LITERATURE REVIEW

With the rapid escalation of the frequency and sophistication of cyber-attacks in this internet era, cyber-security has reached the top priority list. Among all the defensive measures, Intrusion Detection Systems (IDS) have been of great importance to identify and respond to unauthorized network activity. As newer technologies are emerging, IDS frameworks are now moving away from traditional rule-based approaches and opting for machine learning (ML) and deep learning (DL) to enhance detection efficiency and responsiveness.

This chapter presents a systematic survey of IDS literature on the integration of intelligent algorithms for increasing responsiveness and system robustness. A taxonomy of IDS based on detection mechanisms precedes the overview and these are signature-based, anomaly-based, and hybrid-based detection mechanisms. It also explains how ML and DL models such as decision trees, support vector machines, convolutional neural networks, and recurrent neural networks are used to examine complex network traffic and detect anomalies. The literature also mentions adversarial attacks that aim to exploit weaknesses of ML-based IDS and talks about ways to prevent them. Also discussed are specific examples of Distributed Denial of Service (DDoS) attacks, a few of the most disturbing of network infrastructure attacks, and approaches proposed to counteract their impact in intelligent IDS architectures.

2.1 OVERVIEW AND CATEGORIZATION OF IDS TECHNIQUES

Intrusion Detection Systems (IDS) are broadly categorized based on the detection methodology they employ. IDS help to protect computer networks by checking for unusual or harmful activity. This section discusses different types of IDS, including signature-based, anomaly-based, and hybrid systems. Knowing how these systems work makes it easier to choose the right one to keep data and networks safe from cyber threats. Figure 2.1 illustrates a typical intrusion detection system (IDS), where the solid lines show the flow of data and control, while the dotted lines represent the system's response to any detected intrusions. As outlined by the author, a general IDS architecture consists of several important components:



Audit Data Collection gathers data from system logs, network records, and other sources; Audit Data Storage holds this data for analysis and future use; Analysis and Detection applies algorithms to identify intrusions, typically using signature-based, anomaly-based, or hybrid methods; Configuration Data controls system settings; Reference Data maintains known attack patterns or behavior profiles; Active/Processing Data stores incomplete attack information; and the Alarm System manages alerts and responses. Research often focuses on improving analysis and detection, with each detection method having its strengths and weaknesses: signature-based detection identifies known attacks but struggles with new ones, anomaly-based detection finds unknown threats but may produce false alarms, and hybrid detection combines both methods for a more comprehensive approach.

2.1.1 Signature-Based IDS

Signature-Based Intrusion Detection Systems (IDS) work by matching observed network activity against a database of predefined attack signatures. If a known pattern is detected, an alert is triggered. Tools like Snort and Suricata commonly use this method. Signature-based IDS are effective in quickly identifying known threats and are easy to configure. They also generate fewer false positives compared to other methods. But it is not able to detect new or unknown attacks, as they rely entirely on existing signatures. Regular updates to the signature database are essential to maintain their effectiveness against evolving cyber security threats. However, they lack the ability to detect novel or zero-day attacks and require

continuous updates to their signature databases, limiting their adaptability in dynamic threat environments [51].

Roesch [52] introduced Snort, an open-source, rule-based network intrusion detection system capable of real-time packet logging and traffic analysis. The tool demonstrated high efficiency in detecting known threats and offered extensibility through custom signature definitions. A conceptual representation of Snort's detection is given as follows:

$$Alert = \{ 1 \text{ if } P \in R \quad (2.1)$$

Here, P is observed network packet or traffic pattern, R is set of predefined rules (signatures). $Alert = 1$ indicates a detected intrusion, $Alert = 0$ then indicates no match found. Equation 2.1 expresses that Snort raises an alert if and only if an observed packet matches one of the known rules in its database. It also highlights the limitation. If $P \notin R$ the packet represents a new or obfuscated attack not captured in rules, it will go undetected. However, its dependency on pre-defined rules restricts its efficacy against unknown or obfuscated attacks, highlighting its vulnerability to zero-day threats.

Paxson [53] proposed Bro, signature-based IDS that employs an event-driven architecture to analyze network traffic. Unlike traditional IDS, Bro incorporated a scripting language that allowed for flexible specification of detection policies. To formalize the core operation of Bro now known as Zeek, a symbolic equation can be defined that reflects its event-driven and policy-script-based detection approach, while acknowledging its dependency on predefined patterns. Equation 2.2 encapsulates Bro's flexible yet signature-dependent approach. It translates traffic into high-level events and matches them against a set of scripted policies. However, since S is largely based on predefined or manually crafted conditions, Bro shares limitations similar to Snort when facing zero-day or novel attack vectors.

$$Alert = \{ 1 \text{ if } E(P) \in S \quad (2.2)$$

Here, P represents observed network packet or connection event. ' $E(P)$ ' is event generated from packet P through parsing/analysis. ' S ' is set of detection scripts or policies (user-defined or default). $Alert = 1$ if the event matches a condition in ' S '. $Alert = 0$ otherwise. Even though the protocol is extensible but it relied significantly on predefined patterns for detection, limiting its capabilities against evolving and previously unseen threats.

Lau et al. [54] conducted a comparative survey on signature-based and anomaly-based IDS techniques. The study provided a critical analysis of tools like Snort and Suricata, concluding that signature-based IDS perform well in detecting known attacks with minimal false positives. Authors emphasized the inherent limitation of such systems in adapting to emerging attack vectors, necessitating frequent updates to maintain effectiveness.

Ning and Cui [55] provided a comprehensive overview of various intrusion detection techniques, including signature-based approaches. Their work discussed methodologies for signature generation, matching, and deployment in network environments. Authors acknowledged that although signature-based systems are effective for well-documented threats, their major drawback lies in the high maintenance overhead and poor performance against stealthy or adaptive attacks that do not match existing signatures. These studies collectively highlight the predictability of signature-based IDS against well-known threats but also state their drawbacks in aspects of adaptability, scalability, and detection against zero-day attacks. This has given rise to further studies of hybrid and smart IDS architectures leveraging anomaly detection and machine learning towards broader threat coverage.

2.1.2 Anomaly-Based IDS

Anomaly-based Intrusion Detection Systems (IDS) learn and model the normal behavior of a system and later identify any deviation from the baseline as a potential intrusion. Anomaly-based IDS systems are ideal for detecting unknown attacks, including zero-day exploits. However, their main limitation lies in high false positive rates, as normal behavior in many environments is diverse and constantly evolving. Techniques such as statistical profiling, clustering, and machine learning are widely employed to model the baseline behavior. Recent research has focused on enhancing the performance and adaptability of anomaly-based IDS through advanced machine learning and deep learning techniques [56], [57].

Caville et al. introduced Anomal-E, a self-supervised anomaly detection system leveraging Graph Neural Networks (GNNs). By modeling network flows as graphs, Anomal-E captures complex relationships in network traffic without relying on labeled data. This approach enhances the detection of novel attacks. Anomal-E models the network as a graph $G = (V, E)$ where V represents nodes (i.e., ports, IPs). ‘ E ’ represents edges i.e. communication flow.

The core operation of the model can be given as in Eq. 2.3 and Eq. 2.4. Anomal-E leverages structural embedding and reconstruction loss for unsupervised anomaly detection [58].

$$Y = f_{GNN}(G) \quad (2.3)$$

$$Anomaly\ Score = Y - Y_{reconstructed} \quad (2.4)$$

Where, ' f_{GNN} ' is a model that learns latent representations. ' Y ' represents predicted (or embedded) graph features. ' $Y_{reconstructed}$ ' is reconstructed/expected behavior under self-supervised training. Large anomaly scores imply deviation from learned norms. However, the computational complexity of GNNs may pose challenges for real-time deployment in high-speed networks.

A recent study proposed a deep learning-based anomaly detection system tailored for IoT networks. Utilizing a filter-based feature selection and deep neural networks (DNN), the system addresses the unique challenges of IoT environments. The incorporation of Generative Adversarial Networks (GANs) to balance class distributions improved detection accuracy [59]. GAN-DNN enhances supervised learning with synthetic data to address class imbalance in IoT networks. Data generation is given in Eq. 2.5 and classification is given in Eq. 2.6.

$$X_{Aug} = G(z) \quad ; \quad z \sim (0,1) \quad (2.5)$$

$$Y = f_{DNN}(x) \quad ; \quad x \in \{x_{real}, x_{aug}\} \quad (2.6)$$

Here, ' $G(z)$ ' is Generator function of GAN that synthesizes new data to balance class distribution. ' f_{DNN} ' symbolizes deep neural network trained to classify normal vs. anomalous traffic. ' x_{real} ' is Original IoT traffic data. ' x_{aug} ' is GAN-generated synthetic data. ' Y ' represents predicted class label (anomaly or normal). The reliance of model on synthetic data generation sometimes introduces biases, and therefore model's performance in diverse real-world scenarios requires further validation [60].

Teena and Sharma developed a hybrid deep learning model combining ResNet50 and 3D Convolutional Neural Networks (CNNs) for anomaly detection. The integration of Black Widow Optimization (BWO) for weight tuning and Principal Component Analysis (PCA) for feature selection enhanced detection accuracy across multiple datasets [61]. Equation 2.7

shows the hybrid model's data flow and the enhancement mechanisms (PCA + BWO) used to boost detection accuracy where ' $\hat{Y}$ ' represents the predicted label i.e., anomaly or normal).

$$\hat{Y} = f_{3D-CNN} (f_{ResNet50} (PCA(X; k)); \theta_{BWO}) \quad (2.7)$$

Here, ' X ' is the input data (e.g., video frames, network data with spatial-temporal features). ' $PCA(X; k)$ ' is feature-reduced input using Principal Component Analysis retaining k components. ' $f_{ResNet50}$ ' is network which is used for feature extraction. ' f_{3D-CNN} ' is 3D Convolutional Neural Network capturing spatial-temporal patterns. ' θ_{BWO} ' is Optimized weight parameters obtained using Black Widow Optimization. PCA reduces dimensionality for computational efficiency. ResNet50 extracts deep spatial features. 3D CNN captures temporal dynamics. BWO optimizes model weights to enhance classification performance. The model's complexity may hinder scalability and real-time application even if it is effective for network.

Lo et al. presented E-GraphSAGE, a GNN-based IDS designed for IoT networks. By capturing both edge features and topological information, the system effectively detects anomalies in flow-based data [62]. The Embedding of node v at layer k is shown in Eq. 2.8.

$$h_v^k = \sigma (W^k \cdot AGG^k (h_u^{k-1}, e_{uv} \mid u \in N(v))) \quad (2.8)$$

Here, embedding of neighbor node u from previous layer is represented by h_u^{k-1} . Edge feature between node u and v (e.g., packet counts, flow statistics) is represented by e_{uv} . $N(v)$ is the Set of neighbors of node v . Aggregation function (e.g., mean, max, LSTM-based) is depicted by AGG^k . Learnable weight matrix at layer k is denoted by W^k . ' σ ' is the Non-linear activation function (e.g., ReLU). After k layers of message passing, node or graph embedding are passed through a classifier.

$$y_v = Classifier (h_v^k) \quad (2.9)$$

Here ' y_v ' indicates whether the flow or node is anomalous. E-GraphSAGE uses edge-aware message passing, which is crucial for flow-level anomaly detection in IoT networks. It requires data to be structured as a graph with edge features, which limit its applicability in environments where such representations are not readily available.

Talukder et al. proposed a hybrid model combining machine learning and deep learning techniques for intrusion detection. The use of Synthetic Minority Over-sampling Technique (SMOTE) for data balancing and XGBoost for feature selection resulted in high detection accuracy. However, the model's reliance on specific datasets may affect its generalizability to other network environments. Feature selection process is depicted in Figure 2.2. The diagram begins with a list of features $F_1, F_2, F_3 \dots \dots F_N$. The XGBoost algorithm is applied to this feature set to determine the importance of each feature based on its contribution to the model's predictive performance. The resulting feature importance scores are visualized in a bar chart. The process then initializes with $k = N$, meaning it starts by considering all available features. The top k features with the highest importance scores are selected, and a classifier is trained using only these features. The accuracy of the classifier is evaluated using one or more classification algorithms $C_1, C_2, C_3 \dots \dots C_m$ and the resulting accuracy $ACC(k)$ is compared with a predefined threshold $ACC_{thres.}$. If the accuracy meets or exceeds the threshold, the current set of top k features is stored as a candidate feature set. The value of k is then reduced by 2, and the process repeats with fewer features. This loop continues until k reaches zero, at which point the process is terminated. The goal of the iterative process is to reach the minimum set of features but still ensure acceptable classifier performance, thus facilitating model efficiency and accuracy [63].

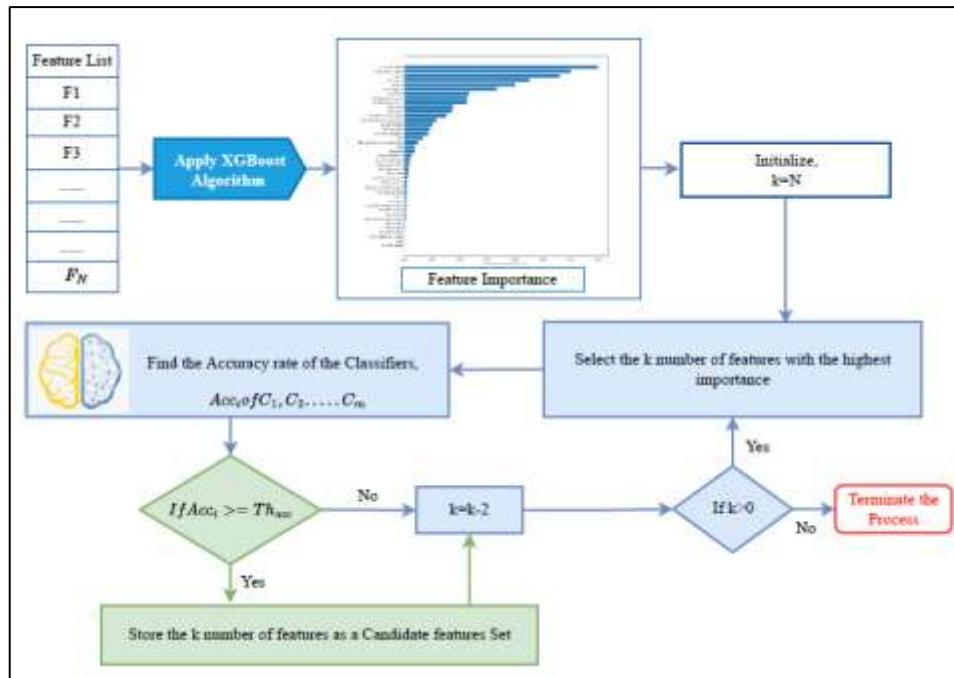


Figure 2.2: Feature Selection Process [67]

These works highlight the ability of sophisticated machine learning and deep learning algorithms in enhancing anomaly-based IDS. A lot has been achieved indeed, but computational cost, scalability, and handling heterogeneity in the network environment remain areas of research challenge.

2.1.3 Hybrid Intrusion Detection Systems (IDS)

Hybrid Intrusion Detection Systems (IDS) combine the signature-based and anomaly-based detection techniques to apply the strengths of both worlds. Signature-based detections excel in identifying well-known attacks with low false positives, while the anomaly-based detections excel at identifying unknown or zero-day attacks. Hybrid IDS attempt to have better detection rates and coverage by combining both techniques but it is challenging to design effective hybrid frameworks since integration is difficult and is computationally costly.

Recent studies have emphasized creating hybrid IDS that are highly effective in balancing detection performance and system efficiency.

Figure 2.3 shows an example of a hybrid system utilizing two basic modules: fuzzy logic to detect known attacks (misuse detection) and neural networks to detect uncommon behavior (anomaly detection). Both modules plug their outputs to a central engine that analyzes both and makes the final decision. This model works well and doesn't need constant updates to its rules.

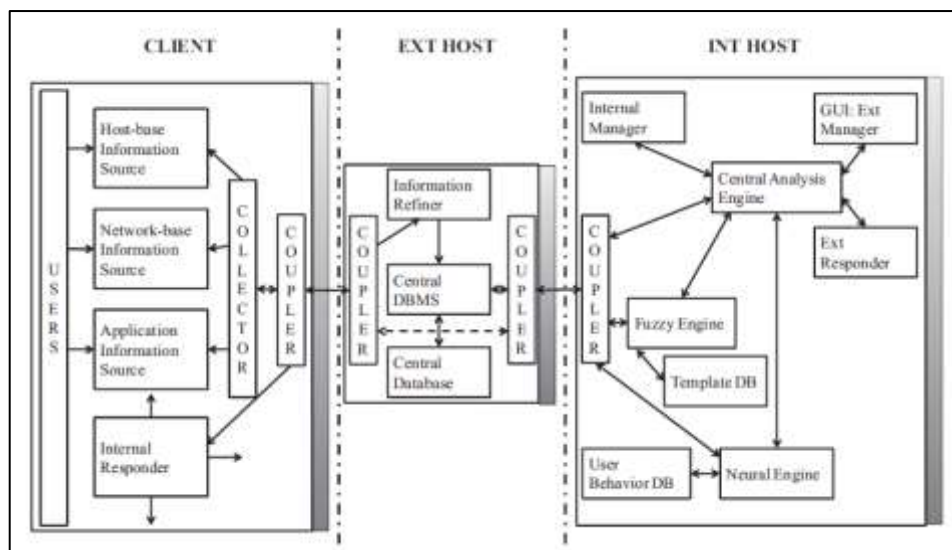


Figure 2.3: Architecture of Hybrid IDS [56]

Li et al. proposed HDA-IDS, a hybrid intrusion detection system tailored for Internet of Things (IoT) environments. The system employs a stacking technique combining multiple classifiers XGBoost, GBDT, AdaBoost, Random Forest, and Extra Trees—for signature-based detection. For anomaly detection, it integrates a semi-supervised Conditional Generative Adversarial Network (CL-GAN) with CNN-LSTM architecture. This dual approach enhances the detection of both known and unknown Denial-of-Service (DoS) and botnet attacks. However, the complexity of the model may pose challenges for real-time deployment in resource-constrained IoT devices [59].

A recent study introduced four hybrid IDS models designed for satellite-terrestrial integrated networks. Each model utilizes Sequential Forward Feature Selection (SFS) based on Random Forest to identify significant features, followed by classification using machine learning or deep learning algorithms such as Random Forest, Long Short-Term Memory (LSTM), Artificial Neural Networks (ANN), and Gated Recurrent Units (GRU). The models were more accurate for task classification in ground network and satellite data sets. However, simulated data set application is likely to limit generalizability to field-based contexts [64]. A hybrid Auto-encoder (AE) and Random Forest (RF) IDS was proposed to achieve better detection capability with less false alarms. RF module is employed to classify pre-classified samples as attacks to check anomalies using reconstruction errors. Two-stage mechanism achieves the best tradeoff between detection power and false alarms. The system performance for dynamic networks with changing attack patterns should be explored [65]. Akif et al. proposed a voting-based ensemble intrusion detection system for IoT networks utilizing four machine learning classifiers Random Forest, XGBoost, K-Nearest Neighbors, and AdaBoost in the ensemble model. Ensemble hybrid was better than individual classifiers for binary and multi-class intrusion detection in case of IoT-23 dataset [66]. While ensemble technique offers maximum accuracy, over computationally complex it may limit scalability for large-scale IoT implementations.

Alharbi and Khan recommended Ensemble Defense System (EDS) as a signature-based and anomaly-based detection hybrid IDS. Elasticsearch is used in the system in real-time monitoring, alarming, data evaluation, and visualization of information. Alharbi and Khan's suggested EDS showed good performance in terms of detection of high rates of cyber-attacks like port scan, privilege escalation, and DoS attack. Utilization of various tools together with

utilization of Elasticsearch can increase the complexity of system configuration as well as maintenance, however [67].

These articles cite the potential of hybrid IDS to strengthen network security through the dual benefit of signature-based and anomaly-based solutions.

2.2 MACHINE LEARNING AND DEEP LEARNING IN IDS

Machine learning has been widely applied in IDS for pattern recognition, anomaly detection, and classification.

2.2.1 Traditional Machine Learning Approaches

Conventional machine learning techniques have been extensively used in the construction of Intrusion Detection Systems (IDS) due to their pattern recognition and classification effectiveness. Of these, Support Vector Machines (SVM), Decision Trees (DT), and Random Forests (RF) have shown promise. Support Vector Machines (SVM) are particularly renowned for their effectiveness in high-dimensional space and have proved effective for binary intrusion classification. Yet, they are not much scalable to big data [68]. Decision Trees (DT) offers interpretability and computation effectively. It makes them suitable for real-time intrusion detection. Yet, they can be prone to over-fitting, especially when handling noisy data [69]. Random Forests (RF) improves classification accuracy and generalization as an ensemble of decision trees. It has been utilized extensively in IDS due to its ability in resisting over-fitting and processing unbalanced data [70]. Current research has explored the improvements and hybridization of these traditional algorithms for overcoming their weaknesses and improving the performance of IDS:

Zhang et al. proposed an intrusion detection method that involves the fusion of the enhanced Random Forest and the Synthetic Minority Oversampling Technique (SMOTE) to handle imbalanced data. By combining K-means clustering with SMOTE, the model effectively increased the representation of minority classes, leading to improved detection accuracy. A representative main equation summarizing the ERF model combined with SMOTE is given in Eq. 2.10.

$$\hat{y} = \operatorname{argmax}_{c \in C} \left(\frac{1}{T} \sum_{t=1}^T h_t^*(x) \right) \quad (2.10)$$

$\hat{y}$ is the predicted class label, 'C' is the set of possible classes, 'T' is the total number of decision trees in the enhanced random forest. $h_t^*(x)$ is the prediction of the t^{th} optimized decision tree for input x , trained using SMOTE-augmented data. The SMOTE algorithm works by synthesizing new examples x_{new} for minority class samples.

$$x_{new} = x_i + \lambda \cdot (x_{zi} - x_i) \quad (2.11)$$

Here, ' x_i ' is a minority class instance, x_{zi} is one of the k-nearest neighbours of x_i and $\lambda \in [0,1]$ is a random number. The Eq. 2.10 shows how SMOTE creates new data points to balance the number of samples in each class. The Eq. 2.11 shows how the Enhanced Random Forest uses multiple decision trees to make a final prediction. Integrating Equation 2.10 and 2.11 expresses the main idea of the SMOTE model. It helps to fix the imbalance in the data, and the Enhanced Random Forest uses that balanced data to detect intrusions more accurately. However, the approach demonstrated a significant drop in accuracy when transitioning from training (99.72%) to testing (78.47%), indicating potential overfitting and challenges in generalizing to unseen data [71].

Umar et al. developed a hybrid intrusion detection system that employs a Decision Tree algorithm for feature selection, combining wrapper and filter-based methods. This approach aimed to reduce computational complexity and enhance detection accuracy. While the method showed effectiveness in identifying relevant features, it incurred higher computational time compared to filter-based methods, potentially limiting its applicability in real-time scenarios [72]. Figure 2.4 shows the process of selecting important features using filter methods and then building models. After selecting the features, WEKA's supervised attribute Remove filter is used to create the final sets of selected features for each feature selection method. Majorly used metric in intrusion detection systems are DR (detection rate), classification accuracy and FAR false alarm rate). These metrics can be represented through Eq.2.12, Eq.2.13 and Eq.2.14 as shown below.

$$Detection\ Rate\ (DR) = \frac{TP}{TP + FN} \quad (2.12)$$

$$Accuracy\ (ACC) = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.13)$$

$$False\ Rate\ (FR) = \frac{FP}{FP + TN} \quad (2.14)$$

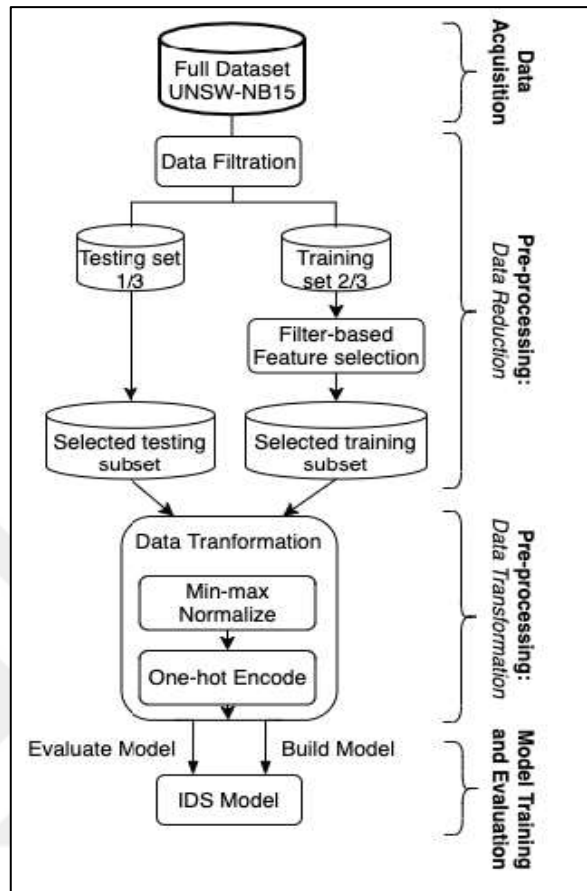


Figure 2.4: Model Training and Evaluation [72].

Dasari and Gottumukkala proposed an intrusion detection system tailored for the Internet of Vehicles (IoV) environment, utilizing an improved Random Forest classifier. The model incorporates optimized feature selection and ensemble learning techniques to enhance detection sensitivity and reduce false positive rates. While the system demonstrated high accuracy in detecting DDoS attacks, its performance in detecting other types of intrusions requires further investigation [73].

Chen et al. introduced an intrusion detection model that combines Adaptive Synthetic Sampling (ADASYN) with Random Forest to address class imbalance in intrusion detection datasets. The model improved classification performance on minority classes, enhancing the detection of less frequent attack types. However, the reliance on synthetic data generation may introduce biases, and the model's effectiveness in real-world scenarios with evolving attack patterns remains to be validated [74].

A recent study introduced a Random Forest-based Network Intrusion Detection System (NIDS) that utilizes permutation feature importance for feature selection. This method aimed to improve detection accuracy and reduce false positives by identifying the most significant features contributing to intrusion detection. Although the approach achieved high performance metrics, its reliance on the CICIDS-2017 dataset may limit its generalizability to other network environments [75]. Figure 2.5 provides an overview of the methodology used to develop a machine learning-based NIDS (Network Intrusion Detection System) using the CICIDS-2017 dataset.

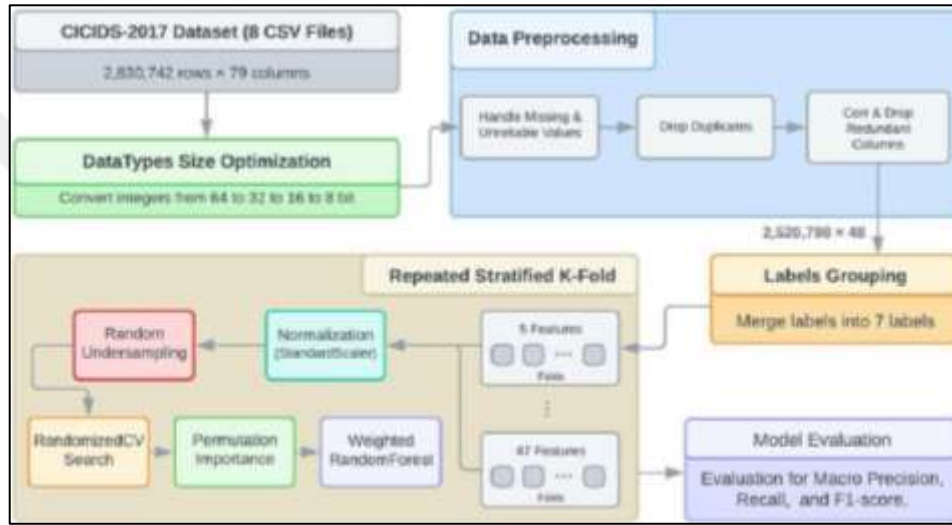


Figure 2.5: Overview of NIDS Model using CICIDS-2017 Dataset [75].

2.2.2 Deep Learning for Intrusion Detection

Deep learning has significantly advanced intrusion detection by automating feature extraction and modeling complex patterns. The following subsections discuss recent developments in various deep learning architectures applied to IDS.

- i. Convolutional Neural Networks (CNN): Yang and Shami proposed a CNN-based IDS utilizing transfer learning and hyper-parameter optimization tailored for IoV environments. The model achieved over 99.25% detection rates on the Car-Hacking and CICIDS2017 datasets. The Eq 2.12 represents simplified CNN-based Prediction with Transfer Learning.

$$\hat{Y} = \text{Softmax}(f_{\theta}(x)) \quad (2.15)$$

Here, $\hat{Y}$ is the predicted class label (e.g., normal or attack type), ' x ' is the input network traffic data (possibly pre-processed and transformed into image-like or tensor form). ' f_{θ} ' denotes the CNN model with parameters θ , partially initialized using transfer learning from a pre-trained model. Softmax function is used to output class probabilities for multi-class classification. Transfer learning is used by initializing the parameters θ of the base Convolutional Neural Network (CNN) from a model that has been pre-trained on a large and diverse dataset, such as ImageNet. This approach allows the model to benefit from learned features and patterns, improving its performance even when applied to smaller or domain-specific datasets like those used in intrusion detection. However, its reliance on specific datasets may limit generalizability to other IoV [77]. A study introduced Rec-CNN, a CNN-based model for intrusion detection in in-vehicle networks. The model demonstrated high accuracy in detecting anomalies but may face challenges in adapting to diverse vehicular network protocols [78].

- ii. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM): Halbouni et al. proposed a hybrid CNN-LSTM model evaluated on the CIC-IDS2017 dataset. The model effectively captured both spatial and temporal features, enhancing detection accuracy. Figure 2.6 depicts IDS model's methodology. Figure 2.7 shows LSTM consists of four gates: forget gate, update gate, tanh gate, and output gate.

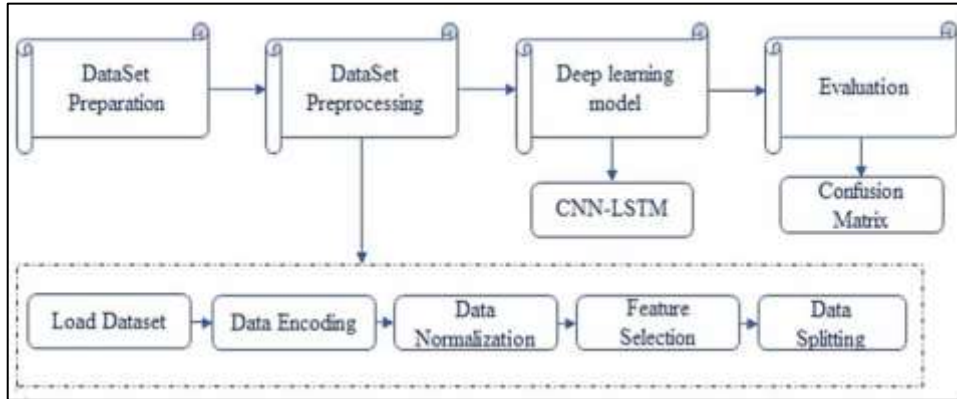


Figure 2.6: Flow of the CNN-LSTM [79].

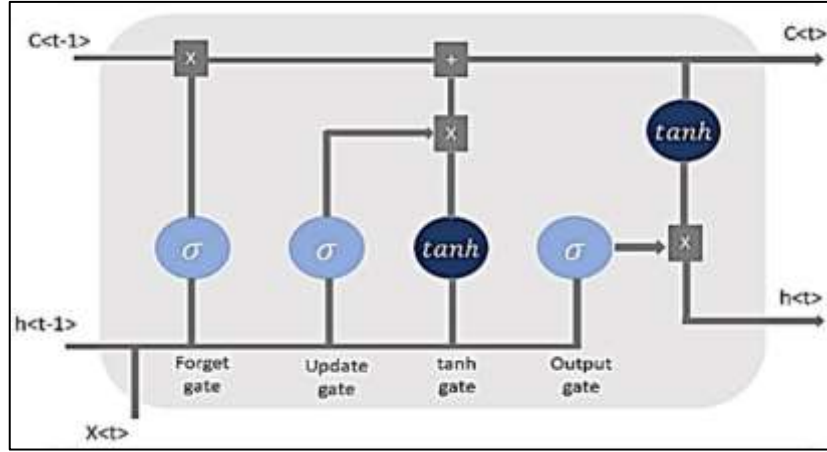


Figure 2.7: Structure of LSTM [78].

But the increased complexity may impact deployment in resource-constrained environments. Saurabh et al. [78] developed LBDMIDS, an LSTM-based model for intrusion detection in IoT networks. The model outperformed traditional ML techniques on UNSW-NB15 and BoT-IoT datasets. The model leverages LSTM's strength in learning from sequential data (like network traffic patterns) to accurately classify benign and malicious activities. The key equations used in an LSTM-based deep learning model are the standard LSTM cell computations, which govern how input data flows through the network. Equation 2.16 decides what information to forget from the previous memory (cell state). Where ' f_t ' is forget gate output at time step t (values between 0 and 1). ' σ ' is sigmoid activation function (squashes values between 0 and 1). Weight matrix for the forget gate is given by ' W '. Concatenation of previous hidden state and current input is given by the term ' $[h_{t-1}, x_t]$ '. ' b_f ' is the bias for the forget gate. If ' f_t ' is 0, it forgets the information. If close to 1, it keeps it.

$$f_t = \sigma (W \cdot [h_{t-1}, x_t] + b_f) \quad (2.16)$$

Forget gate and Input gate equations determine which new information is added to the memory. The equations Eq. 2.17 and Eq. 2.18 together decides what new information to add to the memory.

$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.17)$$

$$\hat{C}_t = \tanh (W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.18)$$

Here, ' i_t ' is the input gate output. ' $\hat{C}_t$ ' is the new candidate values (potential new memory). Weights for input gate and candidate cell state are denoted by W_i and W_c respectively. b_i and b_c are the biases. ' $\tanh$ ' is the Hyperbolic tangent activation function (outputs between -1 and 1). Eq 2.18 updates the cell memory by combining what to forget and what to add. This maintains long-term memory.

$$C_t = (W_c \cdot C_{t-1} + i_t \cdot \hat{C}_t) \quad (2.19)$$

Here, C_t denotes new cell state at time step ' t '. Previous cell state is given by C_{t-1} . The output gate and hidden state equations are given as follows in Eq. 2.19 and Eq. 2.20. These equations together decide what part of the memory to output as the hidden state. Here, Output gate value is denoted by ' o_t '. Hidden state output is given by ' h_t ', Weights for output gate is denoted by ' W_o '. Bias for the output gate is given by ' b_o '. The hidden state ' h_t ' is used in the next time step and for final predictions.

$$o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.20)$$

$$h_t = o_t \cdot \tanh (C_t) \quad (2.21)$$

The prediction layer i.e., the final output converts the LSTM output into a probability distribution over classes. The same is represented using Eq. 2.19.

$$\hat{Y} = \text{Softmax}(W_y \cdot h_t + b_y) \quad (2.22)$$

Here, ' $\hat{Y}$ ' is the final predicted class probabilities. ' W_y ' is the weights of the output layer. ' b_y ' denotes bias of the output layer. ' Softmax ' ensures the output is a probability distribution (sums to 1). In case of binary classification, Sigmoid may be used instead of Softmax. Above equations together allow the LSTM model to understand the sequence and timing of network traffic events. Learn from patterns that unfold over time. Predict whether the current behaviour indicates a normal event or a security threat/intrusion. However, its performance in real-time scenarios requires further validation [79].

- iii. Capsule Networks: A study introduced a few-shot learning approach using prototypical capsule networks with attention mechanisms to address data imbalance in intrusion detection. The method showed promise in scenarios with limited labelled data but may require further optimization for large-scale deployments. Y. Zhang et al. introduced a

protocol which presents few-shot learning approach for intrusion detection using a combination of Prototypical Networks, Capsule Networks, and an Attention Mechanism [80]. The core idea of the method is to classify a query sample based on its distance to class prototypes in the feature space. The main equation is based on the Prototypical Network principle as given below in Eq. 2.20.

$$\hat{y} = \arg \min_c d(f_\phi(x_q), p_c) \quad (2.23)$$

Here, ' $\hat{y}$ ' represents a predicted class label for the query sample. ' x_q ' denotes the query sample to be classified. The term ' $f_\phi(x_q)$ ', denotes embedding (feature representation) of the query sample generated by the Capsule Network with Attention, parameterized by ϕ . ' p_c ' is the prototype vector (mean embedding) of class c , calculated as the average of support examples for class c . Distance function (typically Euclidean distance) used to measure similarity is represented by symbol d .

Wang et al. developed CapsITD, a capsule neural network model for detecting malicious insider threats. The model leveraged graph embedding to capture structural features, outperforming traditional ML methods. However, its applicability to broader intrusion detection tasks remains to be explored [81].

2.3 ADVERSARIAL ATTACKS IN INTRUSION DETECTION

The use of Machine Learning (ML) and Deep Learning (DL) in Intrusion Detection Systems (IDS) has made it easier to detect complex threats. However, this progress has also opened up new ways for attackers to fool these systems. Adversarial attacks aim to trick ML or DL models by slightly changing the data used for training or testing. These small changes can confuse the model, causing it to misclassify threats as normal behavior. As a result, even well-trained IDS models can fail to detect malicious activity. There are three main types of adversarial attacks in IDS. Evasion attacks take place at the prediction or testing phase. Attackers minimally transform malicious data to look normal and avoid detection. Poisoning attacks aim at the training process by adding malicious data to the training dataset, which results in the model learning in an incorrect way. Model inversion attacks and extraction attacks try to reverse-engineer private information from the model or copy the model's behavior by analyzing the model's outputs. It is important to understand how these attacks are executed in order to develop more robust IDS that can protect systems even when

attackers try to manipulate the data. This helps to improve both the security and reliability of intrusion detection systems.

2.3.1 Types of Adversarial Attacks on IDS

Evasion attacks represent a type of adversarial attack that is applied to mislead an Intrusion Detection System (IDS) in the prediction process. In evasion attacks, the malicious inputs are manipulated in small ways to appear normal and become difficult to distinguish from legitimate traffic. Although changes are minor and would otherwise evade human detection, they are specially crafted to exploit weaknesses in the IDS model. Therefore, the system fails to identify the malicious nature of the data and incorrectly identifies it as harmless. This allows attackers to evade detection without triggering any alarms, and hence it is a huge threat to system security. Khan et al. [82] introduced a transfer learning-based adversarial detection model called TAD, which improved resilience against a variety of evasion techniques, although it struggled with cross-dataset generalization. Zhou et al. [83] proposed the NIDS-DA framework to detect functionally preserved adversarial examples in network traffic. While the model performed well under controlled conditions, its robustness declined in the face of novel attack patterns.

Ayub et al. [84] highlighted the practicality of adversarial evasion in real-world scenarios, showing that even minimal perturbations could evade detection. However, the scope of their evaluation is limited to specific vectors. Figure 2.8 shows the effect of adversarial attacks launched on Day n and the subsequent impact observed after the model is retrained and attacked again on Day $n+1$ and Day $n+2$.

Sharma et al. [85] focused on detection and prevention strategies, emphasizing robust feature selection. Yet, their model required higher computational resources. Smith and Wang [86] expanded on practical evaluations of adversarial evasion in IDS, but their exploration remained constrained by limited datasets.

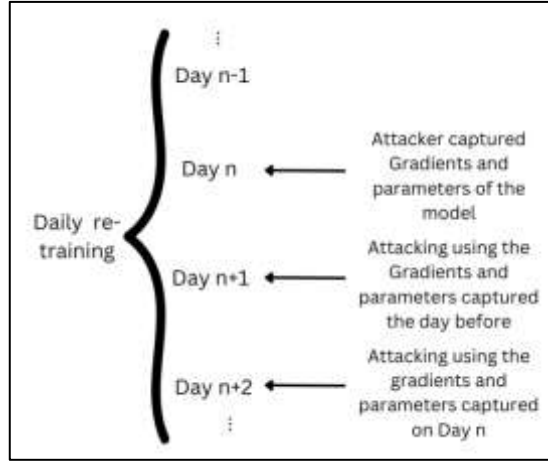


Figure 2.8: Adversarial Attack Scenario with Ongoing Training [87]

Poisoning Attacks: Poisoning attacks compromise the IDS by contaminating the training data with malicious samples, thereby degrading the system's overall performance. Kumar et al. [87] presented a framework to evaluate poisoning attacks in federated learning-based IDS, revealing critical vulnerabilities but lacking validation in centralized systems. Chen et al. [88] applied topological data analysis to detect such attacks, achieving commendable accuracy, though their method is computationally expensive. Zhang et al. [89] analyzed the behavioral responses of different ML models under poisoning scenarios and found ensemble methods to be more resilient. However, their study primarily targeted synthetic datasets. Patel et al. [90] proposed a blockchain-based defense against data poisoning in IDS, which added a layer of security but introduced latency and scalability issues. Liu et al. [91] introduced a personalized federated learning IDS framework with built-in poisoning mitigation, demonstrating promise though requiring broader deployment validation.

Model Inversion and Extraction Attacks: Model inversion and extraction attacks pose a dual threat by attempting to replicate model functionality or extract sensitive data from the training process. Wang et al. [92] presented a thorough survey of inversion attacks to deep learning models, with threats taxonomy and introducing defense methodologies, whereas most of the proposed defense was not implemented practically. Figure 2.9 illustrates how an attacker is able to obtain sensitive input data such as images or feature pattern details by exploiting access to output or internal model parameters of an already existing model. This attack demonstrates the possible privacy risks in machine learning models, especially when models are open or shared through APIs. The attacker reconstructs the original training data by sequential queries or gradients, showing the need for model information protection for

users' privacy. Wang et al. [93] presented HoneypotNet, a novel defense technique that adds backdoors for model extraction attack detection. While useful for entrapping intruders, the use of backdoors is ethically questionable. Other work in this area speaks to growing concern with intellectual property theft and data privacy in leaked models.

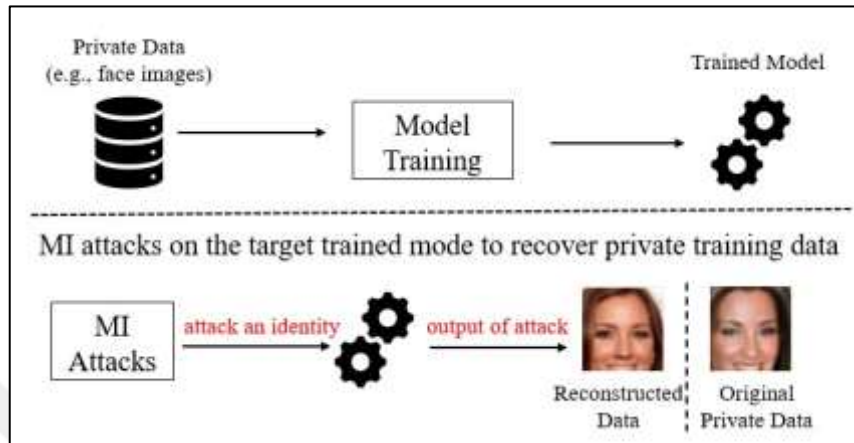


Figure 2.9: Depiction of the Model Inversion Attack Process [93]

2.3.2 Countermeasures Against Adversarial Attacks

Adversarial attacks actually represent a very real threat to the security and integrity of machine learning (ML) systems in such applications as intrusion detection, healthcare, and autonomous systems. Adversarial attacks can actually cause ML models to make bad decisions by manipulating the input data slightly in hard-to-detect directions. In order to counter this issue, researchers have devised several mechanisms of defending ML models and improving their ability to resist such attacks. Adversarial training stands out as one of the most widely used methods where the model is trained on both normal and adversarial examples. This allows the model to learn how to recognize and fight shrewd inputs. Another approach is input pre-processing, i.e., cleaning or pre-transformation of data prior to its submission to the model, so it becomes harder for malicious modification to trick the system. Model regularization and gradient masking are also implemented to reduce the sensitivity of a model for small input changes. Detection mechanisms can also be incorporated to recognize adversarial examples prior to their reaching the principal model. By combining several defense methods, the collective robustness of machine learning models can be improved. Nevertheless, since new attack methods are continually being found by the

attackers, constant research and updating are essential in order to maintain the ML models secure and reliable for real-world use.

Adversarial Training is the most widely used and successful defense mechanism. It involves augmenting the training data with adversarial examples to make the model resistant to similar perturbations during inference. Goodfellow et al. [94] initiated with the discovery of fast gradient sign method (FGSM) based adversarial training. Subsequent research progressed from there. Madry et al. [97] proposed Projected Gradient Descent (PGD) adversarial training, which resulted in much improved robustness against stronger attacks. Eq 2.21 illustrates adversarial training as a min-max problem. In this model, the inner maximization ($\max_{\delta \in S}$) is used to find the worst-case perturbation, i.e., the strongest adversarial example that can be constructed within a set of possible modifications (typically a small perturbation in the input data). This ensures the model is exposed to the most challenging inputs that can mislead it. The outer minimization ($\min_{\theta}$) then focuses on training the model to perform effectively, even when faced with these worst-case adversarial examples. The goal is to enhance the model's robustness by adjusting its parameters to minimize the impact of adversarial attacks, making the model more resilient to future adversarial manipulation. This approach helps build models that are better equipped to withstand adversarial threats.

$$\min_{\theta} E_{(x,y) \sim D} [\max_{\delta \in S} L(f_{\theta}(x + \delta), y)] \quad (2.24)$$

Here, ' θ ' is model parameters to be learned (weights of the neural network). The term ' $(x,y) \sim D$ ' denotes Input-label pairs sampled from the data distribution ' D '. ' δ ' is small perturbation added to the input to create an adversarial example. ' S ' is the allowed set of perturbations (often defined as $\|\delta\|_p \leq \epsilon$). Loss function is denoted by ' $L(\cdot)$ '. The term $f_{\theta}(x + \delta)$ is the model's output for the perturbed input.

Wong et al. [98] developed provable adversarial defenses by incorporating convex outer adversarial polytope bounds. Zhang et al. [99] introduced TRADES, a technique that balances robustness and accuracy by optimizing a trade-off between the two. Rice et al. [100] further investigated early stopping and adversarial training regimes to prevent overfitting to adversarial examples.

Feature Squeezing and Input Pre-processing aim to reduce the input space's degrees of freedom, thereby eliminating unnecessary variations that adversarial perturbations exploit. Xu et al. [95] initially proposed feature squeezing techniques such as reducing color depth and spatial smoothing. Dziugaite et al. [101] explored data compression and quantization as preprocessing defenses. While the authors described research primarily based on empirical and does not introduce new equations, it utilizes the FGSM to create adversarial perturbations. The FGSM computes adversarial examples by adding a perturbation to the original input in the direction of the gradient of the loss with respect to the input. The formula for generating an adversarial example x' from original output x is given in Eq. 2.22.

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \quad (2.25)$$

Here, ' ϵ ' is a small scalar controlling the magnitude of the perturbation. The term ' $\nabla_x J(\theta, x, y)$ ' is the gradient of the loss function J with respect to the input x , given model parameters θ and true label y . The study found that JPEG compression can partially reverse the effects of small-magnitude adversarial perturbations. However, as the perturbation magnitude increases, JPEG compression alone becomes insufficient to mitigate the adversarial impact.

Guo et al. [102] introduced input transformations like bit-depth reduction, JPEG compression, and total variance minimization. Liao et al. [103] applied high-level representation-guided denoising as a defense. Jalal et al. [104] investigated using generative models to purify adversarial examples before feeding them into classifiers.

Ensemble Learning provides robustness by leveraging the diversity among multiple models. The idea is to train several models independently and aggregate their predictions, reducing the vulnerability of the entire system to adversarial manipulation. Liu et al. [96] demonstrated that ensemble adversarial training using multiple models generates more transferable adversarial examples and robust defenses. Pang et al. [105] proposed diverse ensemble models where diversity is explicitly encouraged during training. Tramer et al. [106] incorporated ensemble methods with adversarial training to build defenses that generalize across attacks. Yang et al. [107] developed stochastic ensemble defenses using randomized neural networks. Kariyappa and Qureshi [108] suggested mode connectivity ensembles to improve robustness without significant computational overhead. The core idea of their research is to train ensemble members such that their loss gradients are misaligned,

thereby reducing the transferability of adversarial examples across models. The central component of Diversity Training is the Gradient Alignment Loss (GAL), which quantifies the alignment between the loss gradients of different models in the ensemble. For an ensemble of NNN models, the GAL is defined as in Eq 2.23.

$$GAL = \sum_{i=1}^N \sum_{j=i+1}^N \frac{\nabla_x L_i(x, y) \cdot \nabla_x L_j(x, y)}{\|\nabla_x L_i(x, y)\| \cdot \|\nabla_x L_j(x, y)\|} \quad (2.26)$$

Here, ' $\nabla_x L_i(x, y)$ ' denotes the gradient of the loss function L_i of the i^{th} model with respect to the input x . Symbol ' $\cdot$ ' represents the dot product. Euclidean norm is represented as $\|\cdot\|$. Above equation computes the sum of cosine similarities between the loss gradients of all pairs of models in the ensemble. By minimizing GAL during training, the method encourages the loss gradients of different models to be as orthogonal as possible, thereby promoting diversity. Despite the progress in defensive strategies, achieving universally effective solutions remains elusive. Adversarial tactics evolve continuously, making it difficult for static defenses to maintain long-term effectiveness. Therefore, ongoing research is essential to develop adaptive and resilient systems that can learn and respond to emerging attack vectors.

2.4 DDOS ATTACK DETECTION AND MITIGATION

Distributed Denial of Service (DDoS) attacks are a serious threat in cybersecurity. They rely on sending heavy amounts of traffic from many sources to a network or server with the aim of overwhelming it. It can cause slowing down or completely denying access to legitimate users. DDoS attacks have undergone improvements over time and can avoid standard security solutions as they are based on a multiplicity of sources and have fast-evolving behaviors. These attacks are typically differentiated into three types: volume-based attacks, which flood the network with traffic to use up all the available bandwidth; protocol attacks, which take advantage of the vulnerabilities in the network protocols to use up system resources; and application-layer attacks, which submit fake but plausible requests to a particular app or service. As these attacks are challenging to identify and deter, there needs to be clever security systems that can keep an eye out for odd behavior and respond quickly to continue services running.

2.4.1 Types of DDoS Attacks

Volume-Based Attacks are the most straightforward form of DDoS attacks, aiming to saturate the bandwidth of the targeted network or service. These attacks include UDP floods, ICMP floods, and other high-volume packet floods. These methods attempt to overwhelm the victim's network resources with traffic that exceeds the available bandwidth. Mirkovic and Reiher [109] outlined the fundamentals of bandwidth consumption attacks in early DDoS research. They proposed a comprehensive taxonomy of DDoS attack types and defense mechanisms, offering a structured framework to understand and compare various DDoS strategies and responses. This classification helps researchers and practitioners analyze attack patterns and select appropriate countermeasures. However, the study is majorly theoretical and lacks evaluation through real-world implementation or experimentation. Also, the paper does not address newer, more sophisticated DDoS techniques and evolving attack surfaces.

Wang et al. [112] analyzed traffic characteristics for real-time DDoS detection in high-speed networks. Peng et al. [113] proposed flow-based monitoring systems to detect large-scale floods. Yu et al. [114] presented an entropy-based scheme to capture anomalies in volume-centric attacks. Authors use a hash-based counting algorithm to monitor the number of half-open TCP connections where a SYN is sent but the connection is not completed. The Eq. 2.25 expresses A simplified version of their main detection logic as below.

$$SYN_{rate} - FIN/RST_{rate} > \theta \quad (2.27)$$

Here, ' SYN_{rate} ' is the rate of incoming SYN packets, ' FIN/RST_{rate} ' denotes rate of completed or reset connections. ' θ ' is a threshold value indicating abnormal behaviour.

Luan et al. [115] utilized time-series forecasting models to predict traffic spikes indicative of volumetric DDoS activities.

Protocol Attacks exploit vulnerabilities in network protocols, typically targeting layers 3 and 4 of the OSI model. These include SYN floods, ping of death, Smurf attacks, and fragmented packet exploits. These attacks are often difficult to detect as they mimic legitimate connection requests but never complete the handshake, thereby exhausting the server's connection resources. The classic SYN flood attack, for instance, initiates multiple TCP

handshakes without completing them. Kumar et al. [110] provided an early taxonomy of such attacks. Mirkovic et al. [116] highlighted signature-based weaknesses exploited in protocol designs. The authors introduced a method to trace back DDoS attacks by studying changes in entropy (a measure of randomness) in network traffic. This technique helps identify the source of an attack by spotting unusual traffic behavior. The method may not work well in large-scale or complex networks with high background noise, and it can struggle to accurately trace attacks when the traffic patterns are highly dynamic or encrypted.

Kandula et al. [117] investigated resource depletion from half-open TCP connections. Wei et al. [118] proposed connection tracking to identify anomalies in handshake behavior. Zargar et al. [119] presented a comprehensive survey of protocol-level mitigation strategies.

Application Layer Attacks are among the most challenging to detect because they closely resemble legitimate traffic patterns while targeting application-specific functions such as HTTP, DNS, or VoIP services. Attackers exploit application logic, causing maximum disruption with minimal bandwidth. For example, slowloris attacks maintain multiple HTTP connections by sending partial requests, thereby tying up server resources. Chen et al. [90] discussed the nuances of HTTP-based flood attacks. Xie and Yu [120] proposed browsing behavior models to detect anomalies in user access patterns. Wang et al. [121] explored session-oriented feature extraction for identifying slow-rate DDoS traffic. Wang and Du [122] applied deep learning models to differentiate between bot-generated and human-like behavior in web access logs. Bhuyan et al. [123] used traffic flow clustering for early application-level anomaly detection.

Due to their distributed nature and adaptive strategies, DDoS attacks remain a thorn in the flesh of cybersecurity professionals. Previously employed rule-based detection methods do not seem to be effective in real-time detection of such types of attacks. Therefore, adaptive machine learning-based systems capable of detecting as well as thwarting the advanced threats dynamically are in greater need.

2.4.2 Machine Learning-Based DDoS Detection

Machine Learning (ML) and Deep Learning (DL) techniques have also emerged as powerful tools to detect Distributed Denial of Service (DDoS) attacks by analyzing complex patterns of traffic and marking unusual patterns. Unlike static rule-based systems, ML models

possess the ability to resist evolving attack patterns through learning from key network traffic indicators such as packet size, flow length, protocol, and request frequency. Such models can distinguish good and bad traffic with greater precision if trained on typical datasets [124], [127].

Supervised learning algorithms such as Support Vector Machines (SVM) and Random Forests (RF) have widely been applied in DDoS traffic classification. These classifiers are able to generalize and map the data if provided with labeled sets of data and features that specify discriminative patterns between benign and attack flows. For instance, Kim et al. [128] utilized RF classifiers with flow-based features to achieve high accuracy in detecting both volumetric and low-rate DDoS attacks. It relates to the reconstruction loss of the auto-encoder, which is typically calculated using Mean Squared Error (MSE). The Eq. 2.26 measures how well the auto-encoder can reconstruct normal traffic. A high reconstruction error often indicates anomalous or attack traffic, such as a DDoS attack.

$$L(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^n (x_i - \hat{x}_i)^2 \quad (2.28)$$

Here, ' x ' is the original input vector. ' $\hat{x}$ ' is the reconstructed input from auto-encoder. ' n ' denotes number of input features. The term ' $L(x, \hat{x})$ ' represents reconstruction loss.

SVM-based approaches have demonstrated reliable performance in handling non-linear separating ability in network traffic datasets [129]. Shone et al. [130] implemented an LSTM-based framework that monitored the temporal evolution of flows and flagged suspicious deviations in activity. Temporal models such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are especially effective in identifying sequential patterns and time-dependent anomalies that characterize DDoS attacks. These models can capture historical dependencies in traffic and predict abnormal bursts that may indicate a coordinated attack [131]. GRU models have also been applied to real-time monitoring, showing promise in low-latency environments.

Unsupervised learning techniques, including Autoencoders, offer the advantage of detecting zero-day or stealthy attacks without the need for labeled training data. These models learn compact representations of normal traffic and flag deviations as potential anomalies [132]. Yin et al. used sparse auto-encoders to model the baseline behavior of network traffic and successfully detected outliers corresponding to DDoS events. Clustering algorithms such as

DBSCAN and k-means have also been used to group similar traffic patterns and identify outliers as anomalous flows [133], [134].

Even with all the progress, spotting low-rate or hidden DDoS attacks is still very difficult. These attacks act like normal traffic and stay below the usual warning levels, so they are hard to tell apart from real user activity [135]. To deal with this, researchers are working on smart methods like adaptive learning, choosing the right features, and using a mix of models to make machine learning systems better at detecting such attacks [137].

2.5. IDS IN SDN AND SDNET ENVIRONMENTS

As network systems grow in complexity and shift toward programmable infrastructure, Software Defined Networking (SDN) has become a prominent solution offering both flexibility and scalability in cyber security implementation. By separating the control and data planes, SDN supports centralized traffic oversight and the dynamic application of security rules. This architectural design makes SDN particularly suitable for deploying advanced Intrusion Detection Systems (IDS), especially those utilizing deep learning or hybrid methodologies. Additionally, SDNet an enhanced paradigm that merges SDN with deep learning and edge/cloud technologies provides new avenues for creating adaptive and real-time threat detection frameworks

Extensive studies have investigated integrating IDS within SDN ecosystems to improve accuracy, scalability, and adaptability. These systems take advantage of SDN's network-wide visibility and programmability to facilitate real-time analytics, traffic rerouting, and autonomous response mechanisms. Table 2.1 presents a selection of key studies that incorporate IDS in SDN or SDNet settings, categorized by their detection models, learning methods, system architectures, and primary innovations.

This classification demonstrates the broad range of strategies developed to improve IDS functionality within SDN networks. Starting from early rule-based systems to more sophisticated hybrid models integrate LSTM and GANs, research steadily evolved toward embedding intelligent learning in programmable networks. Many frameworks utilize techniques such as feature engineering, ensemble learning, and staged processing to enhance detection precision and system scalability.

Real-time performance, resilience against adversarial inputs, and adaptability to varied network configurations remain major concerns. Moreover, the potential of combining SDN's programmable nature with the learning capacity of deep models is not fully realized. The SDNet framework seeks to bridge this gap by deploying deep learning units at the controller or edge, supporting continual learning, rapid adaptation, and distributed threat handling.

Table 2.1: Representative IDS Frameworks for SDN and SDNet Environments

Author(s)& Year	IDS Type	ML/DL Technique	Architecture	Key Contribution
Mehdi et al. (2011) [139]	Rule-based IDS	Statistical Rules	SDN Controller-Based	Pioneering IDS for SDN using flow data for anomaly detection.
Braga et al. (2010) [140]	Anomaly-based	Decision Trees	SDN/OpenFlow Switch-Level	Detected DDoS attacks using entropy and flow variance metrics.
Niyaz et al. (2016) [141]	Anomaly-based	Deep Learning (DL)	Controller + DL Classifier	Designed a deep autoencoder IDS tailored for SDN systems.
Tang et al. (2019) [142]	Hybrid IDS	CNN + LSTM	SDNet + Edge Deployment	Combined CNN and LSTM at SDN edges for live threat detection.
Ghosh et al. (2020) [143]	Hybrid IDS	LSTM + Feature Fusion	Controller-Centric Model	Leveraged deep learning for unified traffic and threat classification.
Li et al. (2021) [144]	Adaptive IDS	GAN + LSTM	Federated SDNet Architecture	Employed GAN-based traffic generation for adaptive intrusion detection.
Alharbi et al. (2022) [145]	Signature + Aomaly	Ensemble (RF, SVM)	SDN Controller with SIEM	Created a hybrid IDS using ML and signature-based analysis.

From early statistical techniques to advanced hybrid models using LSTM and GANs, the research community has progressively moved toward integrating intelligent learning systems into programmable networks. Several of these frameworks incorporate feature selection, ensemble classification, or multi-stage analysis to improve scalability and accuracy.

Most existing IDS for SDN environments struggle with real-time processing, adversarial resilience, and generalization across network topologies. Moreover, very few studies fully exploit the synergy between SDN programmability and deep learning adaptability. The concept of SDNet attempts to address this gap by embedding deep learning modules within the SDN controller or edge nodes, enabling continuous learning, real-time adaptation, and collaborative threat response [146].

2.6 SUMMARY AND RESEARCH GAPS

Machine Learning and Deep Learning help computers learn from data. They can find patterns and detect if something is wrong. Many studies show that ML and DL can help IDS to find attacks like DDoS (Distributed Denial of Service), port scan attacks, and zero-day attacks (new attacks that are not known before). These systems use different methods like supervised learning, unsupervised learning, and hybrid methods. These methods help to detect bad network traffic.

But even with many good studies, there are still problems. These problems make it hard to use IDS in real-world networks. In this chapter, main problems are discussed that researchers have found. Major gaps are underlined below

i. Problem with Adaptability

One big problem is that many IDS systems cannot adapt. This means they cannot change or learn new types of attacks after training. Most IDS are trained on old data. When new attacks come, they do not work well. The accuracy goes down. Some researchers talk about solutions like online learning and transfer learning. Online learning helps the system learn while it is running. Transfer learning helps by using knowledge from one task to help another task. But these methods are not used widely. This is because they need a lot of computer power and memory. Also, it is hard to find new and updated data all the time [105], [106], [113].

ii. Problem with Imbalanced Data

Another problem is data imbalance. In real networks, bad traffic (attacks) is very less compared to good traffic. This creates a problem. Machine Learning systems see more good data and less bad data. So, they learn more about good data and not enough about attacks. Because of this, the system gives many false negatives. This means it says there is no attack

when there is actually an attack. Some people try to solve these using methods like SMOTE which creates more corrupt samples, cost-sensitive learning which gives more importance to bad data, and ensemble methods (which use many models together). But these methods are still not perfect, especially when used in real-time on large networks [71], [49].

iii. Problem with Real-Time Use

Deep Learning models like CNN, LSTM, and Auto-encoders are very good. They find attacks with high accuracy. But they need a lot of resources like fast processors and memory. Because of this, they are hard to use in real-time or at the edge (like on routers or firewalls).

Some researchers are working on lightweight models. These models are smaller and faster. Others try to use special hardware like FPGA or GPU. These helps make the system faster. But still, we don't see many systems like this in real use [78], [126].

iv. Problem with Adversarial Attacks

Another big challenge is adversarial attacks. These are smart attacks where bad people change the traffic just a little so that IDS cannot detect it. These small changes are not easy to see. The ML or DL model thinks it is normal traffic, but it is actually bad. This is very dangerous. Researchers are trying to solve these using methods like adversarial training (training the system with smart attack samples), defensive distillation (making models more secure), and input sanitization (cleaning the input data). But till now, there is no strong and complete solution for this problem [82], [85].

v. Problem with Integration in New Networks like SDN

There is also a new kind of network called Software Defined Network (SDN). In SDN, we can control the network using software. This is better than old networks. It helps in fast and flexible control. SDN is good for security systems because it can quickly change how it works when it sees an attack. But only few studies have used Machine Learning IDS with SDN. This is a good chance for future research. We can make smart IDS that work inside SDN and SDNetc (Software Defined Network-centric systems). These can use central control and data from all parts of the network to find attacks faster and better [146].

Because of all these problems, we need to build a better Intrusion Detection System (IDS). This new IDS should be strong and smart. It should work in real-time and should not need

too much computer power. It must use deep learning to detect new types of attacks, even when they are hidden or look like normal traffic. The system should also be able to learn while it is running. This means it should keep getting better as it sees more data. Even if there is more good data and less bad data, it should still work well. It must also be safe from smart attacks where hackers try to trick the system [148], [152]. The IDS should work well with new types of networks like Software Defined Networks (SDN) and SDNetc. These networks are modern and easy to control. So, the IDS should be able to change and react quickly when there is a danger. Such smart and fast IDS can help make the internet safer. It can protect people, companies, and systems from cyber-attacks. With better IDS, we can stop hackers and keep data and networks safe. Machine Learning and Deep Learning are helping a lot in Intrusion Detection. Many researchers have shown good results. But there are still many problems that must be solved. The main problems are lack of adaptability, imbalanced data, high resource needs, adversarial attacks, and lack of use in modern networks like SDN [153], [156].

Next two chapters discuss the foundations of thesis by introducing a Multi-Branched Hybrid Perceptron Network (MHHPN) that integrates multiple DL components with SDN for intelligent threat detection. Incorporation of MHHPN with dynamic feature adaptation and multi-instance learning to tackle class imbalance, evolving threats, and scalability issues making it well-suited for SDNet-centric cyber security deployments.

3. PROPOSED METHODOLOGY ONE

3.1 OVERVIEW OF THE PROPOSED MODEL

Over the years, various machine and deep learning approaches have been developed to address the challenges in detecting such attacks. Researchers across the globe have utilized a range of models like Convolutional Neural Networks, ensemble-based methods, and sequence-oriented networks such as RNNs and LSTMs to capture the intricate patterns underlying these threats. While these methods have shown commendable performance in identifying known threats, they face several limitations. These include difficulty in capturing both spatial and temporal attack dynamics, poor adaptability to evolving attack behaviors, and limited generalization capabilities. To overcome these drawbacks, the current study proposes a hybrid, multi-branched model designed to tackle the weaknesses of existing methods. As depicted in Figure 3.1, the architecture integrates several convolutional layers, followed by hierarchical capsule networks, a masking strategy, and a fully connected output layer [157], [162]. The following section highlights the research gaps addressed and the key innovations of the proposed system.

- i. Most of the previous research studies [12], [13] have been aimed at capturing spatial or temporal dependencies of the attack patterns. The proposed method integrates Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and Capsule Networks into one architecture to capture various features of the detected attack patterns. CNNs are responsible for spatial feature extraction, LSTMs take care of temporal dependencies, and Capsule Networks maintain hierarchical spatial relationships. This convergence of architectures exploits the strength of each network type to best utilize overall detection performance and response time.
- ii. The ability to adapt at all times to emerging patterns of attack is also critical in developing an effective and strong intrusion detection system. In this direction, in this setting, a novel dynamic adversarial learning paradigm is presented where an adversarial generator dynamically generates samples of attacks in real time. The generator is adaptively fine-tuned with model learning always in consideration, producing adversarial examples that probe and improve the classifier's robustness against advanced attack methods. Second, the above technique incorporates an automatic feedback loop within it. That is, the model

is able to update and fine-tune itself automatically whenever new attack types are discovered. Therefore, the model still functions and can effectively contend with increased and evolving threats in the future. This repeated optimization process strengthens the model and ensures that it runs optimally in various and non-stationary traffic conditions.

- iii. Evaluation of the reliability and correctness of the prediction results is critical for the development of any machine or deep learning-based solution. Keeping this in view, the performance analysis of the current work is performed on two diverse datasets to confirm that the suggested architecture demonstrates the desired accuracy and generalization abilities. The performance comparison is carried out with current SOTA: state-of-the-art models to confirm the performance gains realized by the combination of three distinct deep learning architectures with a dynamic learning technique.

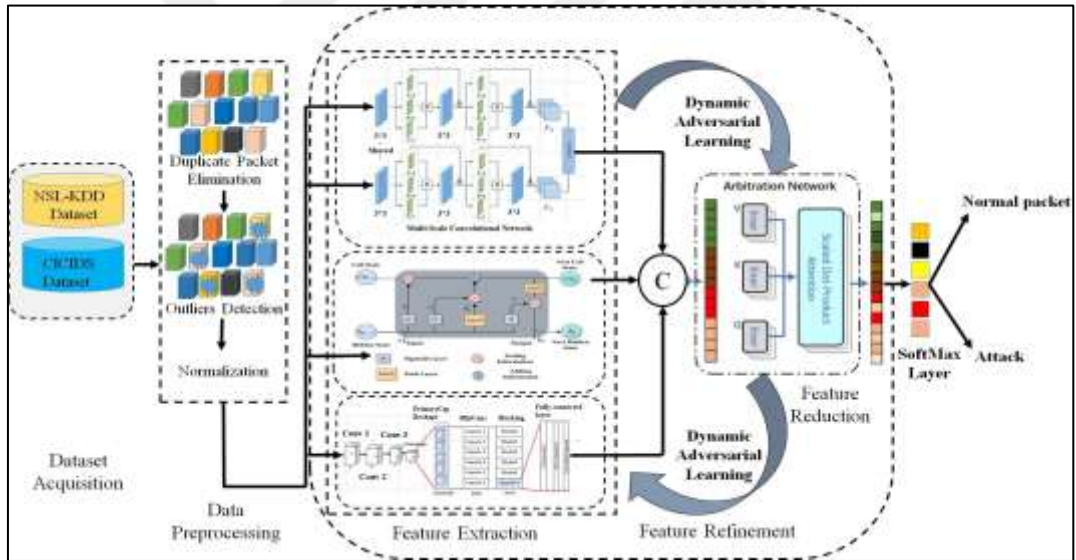


Figure 3.1: Working Methodology of the Proposed Multi Branched Architecture.

The proposed hybrid approach involves amalgamating the strengths of three advanced deep learning architectures, namely Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTMs), and Capsule Networks. Each component in the proposed multi-branched hybrid architecture is designed to address specific aspects of network traffic patterns, thus enhancing the model's overall detection/classification capability. A schematic demonstration of the proposed architecture is shown in Figure 3.2.

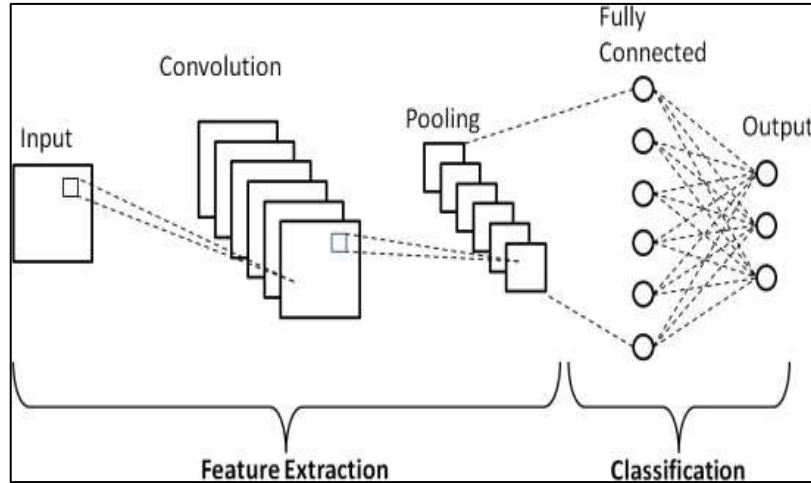


Figure 3.2: Basic CNN Architecture.

3.2 DATASETS FOR INTRUSION DETECTION

In the field of cyber-security, designing and assessing intrusion detection systems (IDS) largely depends on the use of strong, diverse, and well-organized datasets. For the proposed model, which leverages deep learning to identify malicious behavior in network traffic, choosing an appropriate dataset is a crucial step. The main goal is to build highly accurate and efficient deep learning-powered IDS capable of examining traffic and categorizing it as benign or malicious. This classification hinges on the quality of the training data, which should reflect various attack types, balanced class labels, and authentic network patterns. The datasets outlined below are commonly used in both academic and research settings for IDS studies. They offer realistic environments and a wide range of features, making them well-suited for testing deep learning-based detection systems.

3.2.1 NSL-KDD Dataset

The NSL-KDD dataset is an improved and cleaner version of the original KDD Cup 1999 dataset, which has been used as a benchmark for intrusion detection research for years. One of the biggest criticisms leveled against the KDD'99 dataset is the existence of a high number of redundant records that swayed machine learning algorithms in favor of common attack types. The NSL-KDD dataset is proposed to address the problem by deleting duplicate records, resulting in a better sample distribution for different categories of attacks. The database comprises 148,517 records after the pre-processing and cleansing to prevent duplication to the least. The records are split into two sets: about 80% for training and the

remaining 20% for testing. Each record in the database has 42 attributes, which are different attributes like protocol type, service, duration, flag, source bytes, destination bytes, etc., and a label to identify whether the connection is normal or an attack. NSL-KDD divides the attacks into four general categories: DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root), and Probe. These categories mimic a large variety of possible real-world attacks, and thus the dataset is of great value for IDS research. The diversity and richness of NSL-KDD features render it a prime candidate for intrusion detection methods based on deep learning, which can identify complex patterns and outliers in massive datasets.

3.2.2 CICIDS 2017 Dataset

The CICIDS2017 dataset, which the Canadian Institute for Cybersecurity compiles, has become the hip benchmark across recent cybersecurity studies. It is particularly designed to outperform previous datasets by providing a more accurate and comprehensive representation of current network traffic and attacks. Covering five days' worth of data collection, it records normal and attacks traffic, such as Brute Force, Heartbleed, Botnet activity, DDoS, Web-based attacks, and internal network intrusion. One of its strengths is its rich feature design, which derives more than 80 features from raw traffic with the help of CIC Flowmeter. These features derive metrics such as flow duration, packet length, and bytes per second, among others. Every instance of data is manually labeled, so the dataset is well-suited for supervised learning methods in IDS building. It also includes timestamps and flow IDs for every record, which is beneficial for temporal trend analysis, a critical component for time-sensitive deep learning models such as LSTM and GRU. The diversity and size of the dataset make it particularly useful for training models that can identify known and unknown threats.

3.2.3 CIC-DDoS2019 Dataset

The CIC-DDoS2019 dataset is explicitly tailored to DDoS attacks, which are among the most prevalent and unsettling forms of network attacks. The dataset is also created by the Canadian Institute for Cyber-security and is created to better represent a diverse set of DDoS attack vectors in real-world network settings. CIC-DDoS2019 has over 80 million data points, representing over 50 different DDoS attack scenarios, spanning eight general attack classes. The attacks were produced with various botnet tools like LOIC, HOIC, Hulk, and

others, targeting web services, DNS, and other infrastructure. The dataset includes approximately 88 features per record, which are extracted from the network flows and represent detailed information of the attack traffic. These are some of the characteristics that encompass connection time, packets in total, bytes sent and received, and data regarding TCP flags. This makes the dataset highly useful in training models that must learn the particularity of DDoS traffic. Since the dataset is very specific, it is best for model testing geared towards DDoS detection. Deep learning models stand to gain much from this dataset because of the large amount of data and the existence of numerous slight variations between different types of DDoS attacks, which tend to be difficult to identify through traditional approaches.

3.3 DATA PRE-PROCESSING TECHNIQUES

Before feeding any of the datasets mentioned above into a deep learning model, a comprehensive data pre-processing pipeline must be established. This is crucial because raw data often contains inconsistencies, missing values, non-numeric features, and class imbalances that can degrade model performance. Data Cleaning Removal of Redundant Samples: As seen with NSL-KDD, eliminating duplicates is vital to ensure that the model doesn't over-fit to frequent instances. Handling Missing Values: Null or missing entries in datasets like CICIDS2017 and CIC-DDoS2019 are treated using imputation techniques (e.g., replacing with mean, median, or using interpolation). Filtering Noise: Records with anomalous or outlier values are often removed or normalized to reduce their effect on the learning process.

- i. Label Encoding Since most deep learning frameworks require numeric input, categorical features such as protocol type, service, or flag (in NSL-KDD) must be converted to numeric form using label encoding or one-hot encoding.
- ii. Feature Normalization and Scaling Network features often vary widely in range (e.g., packet size vs. flow duration). To ensure stable convergence of the learning process, min-max normalization or standardization (Z-score) is applied to scale features to a common range, typically [0,1] or with zero mean and unit variance.
- iii. Class Balancing Intrusion detection datasets frequently suffer from class imbalance, where the number of normal traffic records vastly exceeds attack records. Techniques such as SMOTE (Synthetic Minority Oversampling Technique), random under-sampling, or data

augmentation are applied to ensure that models do not become biased toward the majority class.

- iv. **Train-Test Splitting** typical split ratio of 80:20 or 70:30 is used to divide the dataset into training and testing subsets. In some cases, a validation set is also carved out to fine-tune hyper-parameters without over-fitting.
- v. **Feature Selection and Dimensionality Reduction** Deep learning models can handle a large number of features, but irrelevant or redundant attributes can still introduce noise. Methods like Principal Component Analysis (PCA), mutual information, or feature importance analysis using tree-based models can be used to reduce the feature space while retaining the most informative variables.
- vi. **Shuffling and Batching** The final datasets are shuffled to remove any order-based bias and then divided into mini-batches to facilitate efficient training of deep learning models using techniques like stochastic gradient descent.

The models become better equipped to learn meaningful patterns and detect a wide range of cyber threats effectively with continuous pre-processing techniques, the. The accuracy of the any learning model is highly affected by the quality of the dataset available for analysis. Thus, data pre-processing is considered as one of the most critical phases in the machine learning pipeline, as it ensures the data is clean, consistent and well-structured for analysis [162]. Moreover, it involves several sub-tasks to remove any inconsistencies or noise present in the data. In context of the proposed work, the following data pre-processing steps have been applied on the considered dataset to optimize its use for deep learning models [163].

3.3.1 Scaling

In machine learning, especially in deep learning models, it is essential to treat all input features fairly during training. One common challenge that arises is the presence of dominant features numerical attributes that have significantly larger values or wider ranges than others. If left unaddressed, these features can disproportionately influence the learning process. It leads to the model incorrectly prioritize certain patterns while ignoring others that might be equally or more important for prediction. To mitigate this issue, feature scaling techniques like min-max normalization are employed. Min-max normalization is a widely used scaling technique that transforms the values of numerical features into a fixed range, typically

between 0 and 1. The scaling technique employed is mathematically defined as follows: The formula for min-max normalization is given in below Eq. 3.1.

$$IDf_{scaled} = \frac{IDf - IDf_{min}}{IDf_{max} - IDf_{min}} \quad (3.1)$$

Where, ' IDf ' is the original feature value. ' IDf_{min} ' and ' IDf_{max} ' are the minimum and maximum values of the feature respectively and ' IDf_{scaled} ' is the normalized value.

By applying this technique, all numerical features are brought to the same scale, effectively neutralizing the dominance of any high-range features. This is particularly important for algorithms that are sensitive to the scale of data, such as neural networks, k-nearest neighbors, or gradient descent-based models. In such models, not scaled features can cause optimization to take longer or converge to suboptimal solutions because features with large values can lead to disproportionately large gradient updates. In the context of the proposed intrusion detection model, which relies on deep learning to classify network traffic as normal or malicious, the dataset contains a wide variety of network features such as duration, number of bytes transmitted, packet sizes, and others. These features naturally exist on different numeric scales for instance, one feature might be in the range of 0–1, while another might span thousands. If these disparities are not addressed, the model could wrongly assign more importance to high-range features like "bytes sent" simply because their numerical magnitude is greater rather than because they are more relevant to identifying attacks. Applying min-max normalization ensures that each feature contributes equally to the learning process, allowing the model to focus on discovering meaningful patterns in the data rather than being skewed by numeric disparities.

In addition, normalization also accelerates the training process by stabilizing the gradient descent steps and minimizing the chances of exploding or vanishing gradients. Min-max normalization is a powerful yet simple pre-processing method that is one of the most significant elements making deep learning models efficient and trustworthy. Since everything is within the same range, it allows the model to make more balanced decisions, learn as required, and carry out tasks like intrusion detection more effectively.

3.3.2 Encoding

Deep learning requires numerical data so that multiple layers can operate on it and perform mathematical operations and learn features. In real data like intrusion detection data in real data, there are some features which are not categorical in nature. They are quantitative but will have or will not have quantitative values but are various groups or classes like network protocols or service types. In order to utilize them in deep learning models, they should be numerical with no spurious ordinal ones. One of the most powerful and popular methods of doing that is one-hot encoding. One-hot encoding substitutes every categorical feature with a binary (0 or 1) column vector, one for each unique categorical value. For example, if there's a feature named "protocol type" with three unique values: TCP, UDP, and ICMP, one-hot encoding would substitute this single column with three separate binary columns, one for the presence (1) or absence (0) of each of the protocols. This is due to the fact that merely calling these categories by this simple naming of numbers (i.e., TCP = 1, UDP = 2, ICMP = 3) would imply a ranked or ordered relation between them that may not necessarily be the case. One-hot encoding prevents this and ensures that the deep learning model treats each category as independent and equally significant. In the proposed deep learning-based intrusion detection model, the dataset includes several categorical features (listed in Table 3.1), such as protocol_type, service, and flag. These fall under the nominal category, meaning they represent labels without any order. Applying one-hot encoding to these features ensures they are accurately represented in a numerical format, enabling the deep learning model to learn meaningful patterns from them.

Table 3.1: Description of the Dataset Utilized in the Present Study [159]

Basic Features	Traffic Features	Content Features
Contain information about the TCP connections such as duration, protocol type, service, connection duration, connection flag and so on.	Contains statistical information about the connections in the same window (surrounding network traffic), number of connections from same host, percentage connections with syn errors, and so on.	Attributes defining the content within a connection such as number of failed login attempts, compromised conditions and many more.
Types of Features		
Nominal (Categorical Features),	Binary Features,	Numeric attributes
Protocol_type, service and flag	land, logged_in, is_guest_login, su_attempted, root_shell, is_host_login	All the remaining attributes are numeric in the dataset.

The dataset's features can be broadly categorized into three main groups: Basic Features, Traffic Features, and Content Features, each offering unique insights into network activity. Basic Features provide fundamental information about each TCP connection. These include attributes such as duration, which represents the total time of the connection; protocol type, which indicates the type of protocol used (e.g., TCP, UDP, ICMP); service, which identifies the network service on the destination (e.g., HTTP, Telnet, FTP); and flag, which represents the status or outcome of the connection (e.g., SF, REJ). These features are essential for understanding the structure and nature of network communication. Notably, protocol type, service, and flag are categorical features and must be converted into a numerical format using one-hot encoding to make them suitable for processing by deep learning models. Traffic Features capture the statistical behavior of the network traffic over a window of time or within a specific context, such as host activity. These include metrics such as the number of connections to the same host, the percentage of connections that have SYN errors, and the number of connections to the same service. These attributes are generally numerical since they are calculated based on aggregations of several connections. They aid in detecting strange patterns and outliers in network behavior and are very useful for discovering distributed or coordinated attacks. Content Features explore the real content and behavior within a single connection. These attributes are particularly significant in discovering signs of intrusion based on internal behavior. Examples are the number of failed login attempts, whether the connection is from a guest login, whether the root shell is logged in, and whether the user is logged in or not. Some of them are binary features, meaning a simple yes or no status. These characteristics enable the capture of more in-depth contextual information that cannot be derived from connection or traffic statistics and are critical in identifying advanced attack patterns like unauthorized access or privilege escalation. All three types of features together give a complete picture of network activity, integrating structural, statistical, and contextual information. Appropriate handling and transformation of these features via techniques like one-hot encoding and normalization are necessary for efficient training of deep learning models for intrusion detection.

3.3.3 Data Splitting

One of the most important steps in preparing a dataset for deep learning model training is to split it into three separate subsets: training, validation, and testing. This split lies at the core

of effective model training, tuning, and fair evaluation. Each subset serves a particular purpose in the development cycle and adds to the overall robustness and generalizability of the model. Training set is utilized in learning the deep model to make it learn patterns, features, and relationships from input. It covers the majority of the dataset and depicts examples which model leverages in learning internal parameters using techniques such as stochastic gradient descent. The validation set, however, is used within the training process to optimize model hyper-parameters, avoid over-fitting, and check performance. It serves as a checkpoint for the model to check how well the model will generalize to unseen data within training.

This makes it possible for early stopping and model selection methods that lead to improved overall performance. The test set is reserved until the model has been trained and validated. It gives a final, unbiased estimate of the accuracy and performance of the model on unseen data. This prevents the reported performance measures from being overly optimistic as a result of exposure during training. In order to provide good results, the splitting process here has to be done with precision. Generally, a 70-15-15 or an 80-10-10 ratio is practiced, where training gets a majority share, and the remaining is evenly divided between test and validation. The choice can be random but stratified in case of class imbalance problems that are common with classification issues like intrusion detection. Stratification preserves the proportionality of both attack and normal classes in all sets. In addition to data splitting, careful data quality checks are necessary. One of the main checks is to detect and manage missing values. Missing values may result in errors while training or make the model learn the wrong patterns. Methods like imputation (with the help of mean, median, or mode) or, in case the proportion is very low, exclusion of the respective records are employed most frequently based on the proportion and severity of missing data. Outlier analysis, i.e., detection of outliers as data values that are far from the rest of the data set, is another distinguishing characteristic.

Outliers disrupt model training by introducing noise or distortion of existing relationships, most particularly in sensitive models. Outliers are identified and measured using different techniques such as Z-score, IQR (Interquartile Range), or graphical tools such as boxplots. Outliers may be deleted or plotted depending on their context and effect on identification. Splitting the data into the training, validation, and test sets is an important step in machine

learning workflows to have an unbiased and fair model assessment. Adding sufficient missing value and outlier checks on top of this makes the dataset clean, standardized, and model-ready. The pre-processing steps make sure that the proposed deep learning-based intrusion detection system will be effective, generalizable, and trustworthy.

3.4 MULTI-SCALE SPARSE CNN FOR SPATIAL FEATURE EXTRACTION

Convolutional Neural Networks (CNNs) are powerful deep networks used extensively to extract complex and meaningful features from structured data. Originally developed for image analysis tasks, CNNs have been utilized extensively in a wide range of domains, such as healthcare, agriculture, and cyber-security. In intrusion detection systems, CNNs are best used to detect spatial patterns in network traffic. The building elements of the basis CNN convolution filters, pooling operation, and flattening operations cooperatively identify salient features out of the given input. To conduct this work, a multi-scale sparse CNN is employed that learns spatial feature of network traffic data. The network itself is particularly designed such that it modifies dynamically its own filters so that it can sense patterns for exceptional behavior in the traffic, i.e., doubtful access attempts or frequent requests. By analyzing these spatial interdependencies, CNN assists the model in identifying potential threats correctly. With the multi-scale and sparse architecture, the model further excels at detecting a broad scope of attack signatures by prioritizing significant features and reducing computational complexity. This mechanism assists the system in sustaining robustness and flexibility against continuously changing attack methods and patterns in network traffic.

Convolutional Neural Networks (CNN) [159], [164] are commonly used and efficient neural models for extracting high-level and complex features in the dataset. Initially, they were commonly used in various application areas, including healthcare, agriculture, and many more. The elementary architecture components of a CNN structure include filters (convolution: to extract intrinsic features), pooling, and flattening layers (as depicted in Figure 2). These fundamental elements are tasked with defining features and patterns of the network traffic dataset. In the case of the work under the present study, the CNN element is designed to adjust its filters in accordance with changing spatial features in the network traffic data [8]. For example, the filters are optimized to extract vital patterns regarding abnormal traffic flows or recurring connection requests. By learning from the patterns in the input data set, the CNN picks up and examines spatial dependencies within the traffic,

enhancing its capability to identify and classify sophisticated attack signatures (Algorithm 3.1: Line 8-14). Such flexibility of CNN filters guarantees that the model is capable of managing variations in attack patterns and is able to identify attack signatures and traffic features effectively.

3.5 GRID LSTM FOR TEMPORAL DEPENDENCY LEARNING

While spatial attributes assist in recognizing structural trends in network traffic, knowing how the traffic evolves is just as crucial. To take these time-dependent (temporal) patterns into account, the suggested model uses Long Short-Term Memory (LSTM) networks. LSTMs are a form of recurrent neural network (RNN) that can remember information over long sequences and are, therefore, better than typical RNNs at identifying long-term dependencies in sequential data. In the context of this work, the LSTM module is responsible for learning time domain behaviors of network traffic, i.e., repeated login attempts or bursts in activity in the long term. To further improve the temporal modeling, a dynamic attention mechanism is introduced, which assists the model in paying attention to the most significant part of the sequence that may be indicative of possible threats.

This focus-based configuration enables the model to dynamically modify its focus based on the way traffic patterns change, rendering the model more responsive to newly emerging or changing attack patterns. In general, incorporating LSTM with temporal attention enhances the capability of the system to analyze time-sensitive features, which leads to better and earlier detection of malicious behaviors. Along with the capture of spatial properties of the dataset, it is important to pay attention to the temporal (time-dependent) dependencies of the network traffic dataset. Sequential models such as RNNs and LSTMs have had tremendous success in extracting and modeling temporal dependencies [165], [166].

Moreover, LSTMs have been proven to be more effective and powerful as they are capable of handling long-term data sequential dependencies as compared to RNNs. Hence, the current work employed LSTM networks (as shown in Figure 3.3) to extract the temporal aspects of the network traffic. For instance, the LSTM network can be adopted to characterize repeated login attempts and timed data bursts. Additionally, a temporal attention mechanism is also added to the basic sequential architecture. This introduction of a dynamic temporal attention mechanism allows the LSTM network to prioritize sequential

dependencies and patterns based on evolving traffic behaviors. This capability is crucial for understanding and predicting temporal variations in the network traffic. Further, this enables the model to adapt to evolving attack strategies over time (Algorithm 3.1: Line 15- 20). On a whole, the introduction of LSTM model allows the hybrid architecture to manage time-dependent features and enhance the accuracy of attack detection.

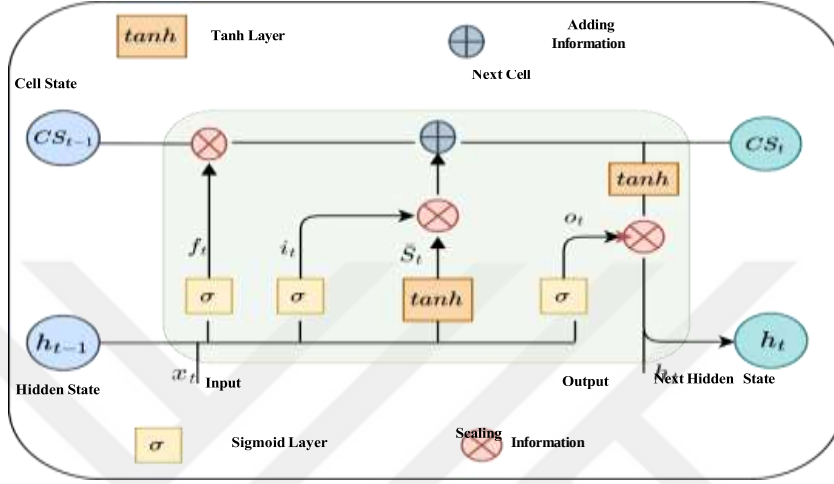


Figure 3.3: LSTM Cell Architecture.

3.6 STACKED CAPSULE NETWORKS FOR HIERARCHICAL FEATURE LEARNING

The capsule network [167] is the third and final branch of the proposed multi-branched architecture. This phase of the proposed approach allows for capturing hierarchical spatial relationship present in the data. Basically, the network allows preserving spatial relationships across features, addressing the challenge of handling complex data variations. Compared to the CNN, these networks allow for capturing detailed and part-to-whole relationships among the specific characteristics of a given object. Capsule Networks maintain hierarchical relationships and spatial configurations within the data, which is essential for accurately classifying diverse and evolving attack patterns. This approach ensures that the model can effectively manage intricate feature interactions and variations in the data (Algorithm 3.1: Line 21-28). The overall architecture of this layer comprises multiple convolution operations followed by hierarchical capsule layers, masking mechanism and the fully-connected layer (shown in Figure 3.1).

3.7 DECISION-LEVEL FEATURE FUSION STRATEGY

The Decision-Level Feature Fusion layer plays a vital role in the proposed hybrid model by bringing together the feature vectors generated from different deep learning components—namely the CNN, LSTM, and Capsule Network. Each of these architectures captures different aspects of the input data: spatial, temporal, and hierarchical features. By combining their outputs, a more comprehensive and informative representation of the network traffic is created. This combined feature vector is then passed through an attention mechanism, which helps highlight the most relevant parts of the data for decision-making. Following the attention stage, the feature representation learned is flattened and passed through dense (fully connected) layers. The probability score is then calculated in the output layer and used to classify the input as normal or an attack. This combination strategy enables the model to take advantage of each of the architectures' strengths and provide both detection performance and its capacity to cope with complex and dynamically changing threats. This layer is significant because it is the concatenation of the output feature vectors of the various architectures (CNN, LSTM, and Capsule Network) utilized in the hybrid model (Algorithm 3.1: Line 29-31). Feature vectors calculated from such networks are concatenated to provide a concatenated representation (as can be observed in Figure 3.4). Then, they are sent to the attention mechanism to specify the key elements. The merging takes advantage of the strength of each type of network to enhance overall detection capability and responsiveness. Finally, the obtained feature representation is flattened and passed to the linear layer (dense layers). The probabilistic representation generated at the model output is utilized to define the class label (normal or attack).

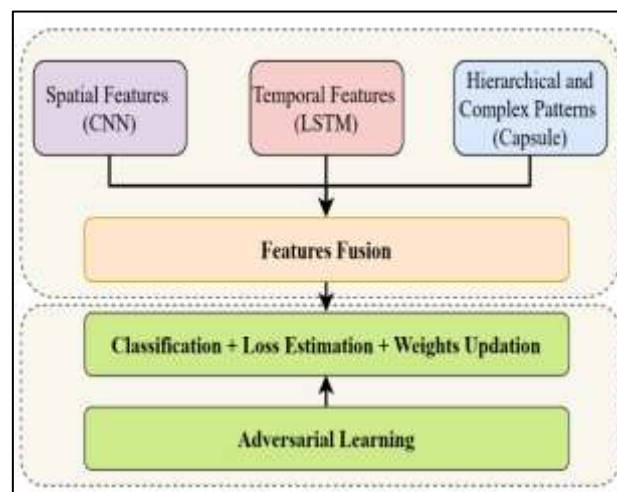


Figure 3.4: Representation of Features Fusion and Adversarial Learning.

3.8 ADVERSARIAL LEARNING WITH THRESHOLD-BASED ATTACK DETECTION

A core component of our proposed multi-branched hybrid approach is the dynamic adversarial learning scheme [168], which further enhances the robustness of the hybrid approach (Algorithm 3.1: Line 32-36). This allows the introduction of a real-time adversarial generator that creates attack samples by identifying weaknesses in the model's decision boundaries. This generator continuously adapts to the model's learning progress, generating adversarial examples that challenge the classifier and test its robustness. By targeting the most vulnerable patterns in the NSL-KDD dataset [159], the adversarial examples ensure that the model remains resilient against sophisticated attack strategies. The architecture also features a dynamic feedback loop that allows the classifier to adjust its parameters in response to evolving adversarial attacks (Figure 4). With the adversarial generator refining its attack strategies, the model learning process adapts to these changing attack patterns. The inclusion of this feedback-driven optimization enhances the classifier's resilience and improves its ability to generalize across diverse and evolving traffic patterns. Furthermore, the figure 3.5 indicates the steps of the proposed model how the attacks pass through the feature extraction.

In a nutshell, the proposed approach integrates dynamic adversarial learning strategy with three deep learning architectures (CNN, capsule and LSTM) to resolve the key challenges associated with network traffic activities identification task. The integration will help in enhancing the model's overall robustness, and adaptability, thus providing a comprehensive solution for attacks detection and mitigation. The performance assessment procedure of the proposed multi-branched network traffic classification approach (Figure 3.6: Line 38-40) has been discussed.

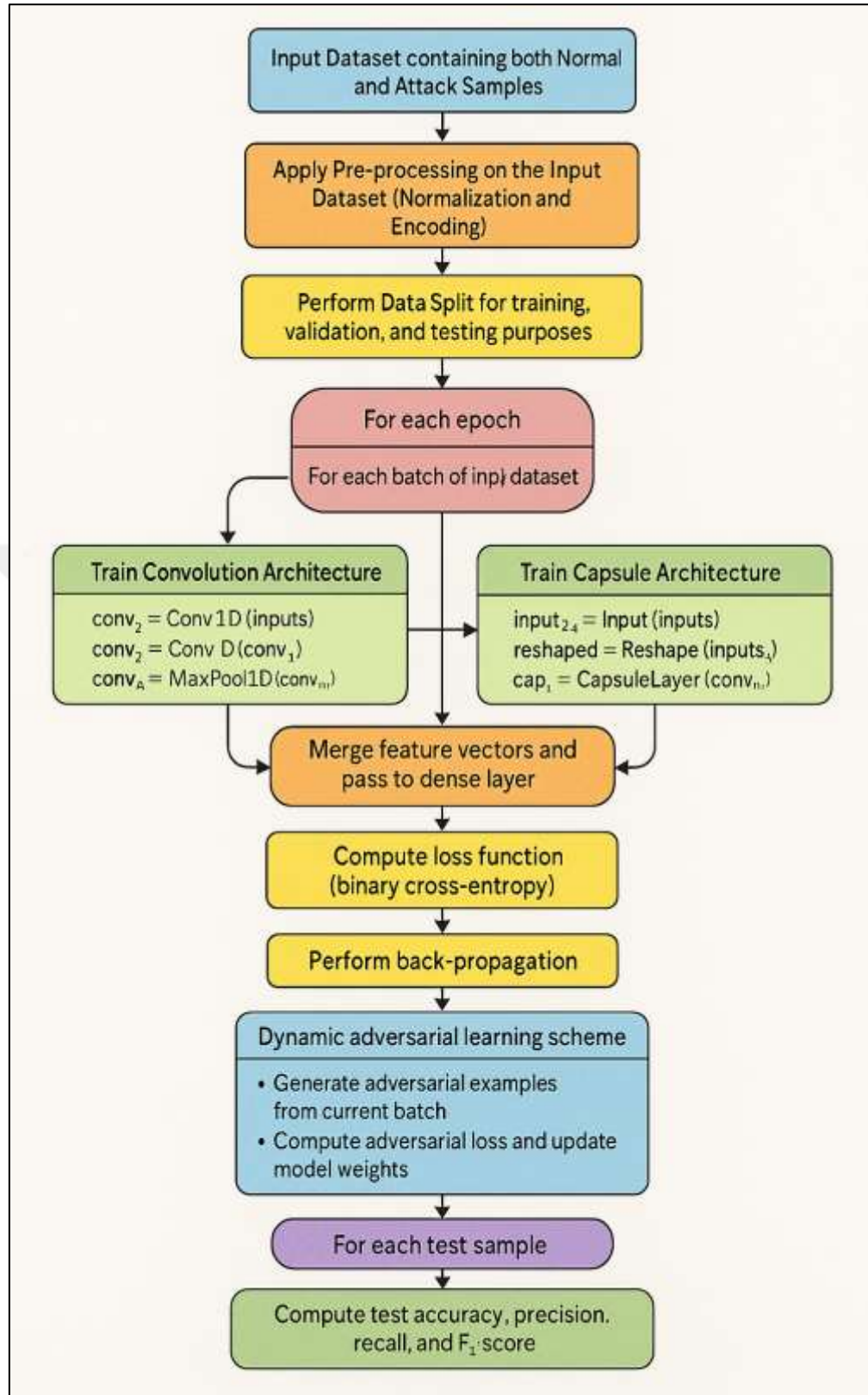


Figure 3.5: Workflow of the Proposed Model.

```

1 Input: Input Dataset containing both Normal and Attack Samples;
2 Output: Proposed Multi-Branched Model
3 Apply Pre-processing on the Input Dataset (Normalization and Encoding)
4 Perform Data Split for model training, validation and testing purposes.
5 for each epoch:
6     for each batch of input dataset:
7         begin
8         Train Convolution Architecture:
9         begin
10             $conv1 \leftarrow Conv1D(Inputs)$ 
11             $conv2 \leftarrow Conv1D(conv1)$ 
12             $conv3 \leftarrow Maxpool1D(conv2)$ 
13            ...
14             $convn \leftarrow Conv1D(convn-1)$ 
15        Train LSTM Architecture:
16        begin
17             $inputs1 \leftarrow Inputlayer(Inputs)$ 
18             $lstm1 \leftarrow LSTM(Inputs1)$ 
19            ...
20             $lstmn \leftarrow LSTM(lstmn-1)$ 
21        Train Capsule Architecture:
22        begin
23             $Inputlayer \leftarrow Input(Inputs)$ 
24             $reshaped \leftarrow Reshape(Inputlayer)$ 
25             $conv1 \leftarrow Conv2D(reshaped)$ 
26            ...
27             $cap1 \leftarrow Capsulelayer(convn)$ 
28            Add more capsule layers, squash operation defining the number of capsule layers and
capsule dimension.
29            Merge the feature vectors extracted from all three architecture and pass it to the dense
layer.

30            Compute the loss function (binary cross-entropy for classification).
31            Perform back-propagation to update weights of CNN, LSTM, and Capsule Network
components.
32            Dynamic adversarial Learning Scheme:
33            begin
34            Generate adversarial examples from the current batch.
35            Pass the adversarial examples through hybrid approach and compute adversarial loss.
36            Update model weights using a combined loss (normal + adversarial loss)
37            Validate the model on the validation set after each epoch.
38        begin
39            For each testsample
40            Compute test accuracy, precision, recall, and F1-score.

```

Figure 3.6: Illustrates the Pseudo-code for Proposed Model.

3.9 SUMMARY

This chapter provides a detailed description of the proposed deep learning-based system for detecting intrusions in network traffic. The main objective of this approach is to identify harmful or suspicious activities that may compromise network security. To achieve this, the

model integrates multiple advanced learning techniques, enabling it to analyze, process, and classify various forms of network data more effectively. The architecture of the model is designed to capture different types of data characteristics, spatial, temporal, and hierarchical by using a multi-path structure.

This framework facilitates more effective identification of advanced and constantly changing cyber-attacks. Three prominent datasets are considered for the model to be trained and tested: NSL-KDD, CICIDS2017, and CIC-DDoS2019. These are most commonly used in the cyber-security community as they consist of diverse instances of normal and attack-based network traffic behaviors. All of these datasets are helpful to make the proposed model's performance more generalized and stronger. Pre-processing of the data prior to input into the model is an important step. Hence, some data pre-processing methods are used. These involve normalizing numerical values with techniques like Min-Max scaling to scale features into comparable ranges, which enhances the performance of the model. In order to deal with concerns associated with uneven data, SMOTE (Synthetic Minority Over-sampling Technique) and under-sampling methodologies are utilized to balance attack samples and normal samples.

Furthermore, feature selection and engineering are performed so that the most pertinent features that directly impact model performance can be located and maintained. The new proposed model consists of multiple deep learning components. A Multi-scale Sparse Convolutional Neural CNN is used first to learn spatial characteristics from network data. It assists in detecting localized patterns, which are usually indicators of an intrusion. Grid-based Long Short-Term Memory (LSTM) network is used second by the model to learn temporal patterns. Network traffic being time-varying, it assists the model in learning attacks built over time or time-varying attacks. To incorporate further learning enhancement, Capsule Networks are stacked to identify hierarchical features of data. These networks work effectively in preserving spatial information among features such that the model can identify small variations in traffic flow, which other methods may overlook. The three-branch output, CNN, LSTM, and Capsule This method integrates the strength of both models and enables the predictions to enhance. A dynamic adversarial learning module is implemented to improve the capacity of the model in dealing with new or unseen attack types. The module learns the model on real and synthetically generated instances of attacks, thereby improving

the model's robustness against sophisticated and new attacks. The entire framework is experimented under a given experimental environment through accuracy, precision, recall, and F1-score evaluations. The performance analysis demonstrates the functionality of the model with different types of intrusions. While the model is contrasted with datasets like NSL-KDD, CICIDS2017, and CIC-DDoS2019, in subsequent studies, more extensive verification can be provided with comparable standard datasets like UNSW-NB15. In total, this chapter introduces an adaptive and efficient intrusion detection system that addresses the flaws of current systems. These networks are combined using the assistance of a decision-level fusion technique. By combining a number of deep learning techniques with the introduction of adversarial learning, the solution presented in this research has enhanced adaptability and performance in identifying different kinds of network attacks. Experimental results supporting these assertions are given in Chapter 5.

4. PROPOSED METHODOLOGY TWO

4.1 INTRODUCTION TO DDOS ATTACK CLASSIFICATION

In the ever-evolving landscape of cyber security, Distributed Denial of Service (DDoS) attacks remain a major threat to digital infrastructures, online platforms, and organizations [169], [170]. Malicious actors often deploy botnets—networks of compromised devices to launch DDoS attacks, enabling control over thousands of systems simultaneously (Figure 4.1). These attacks typically involve overwhelming targets with excessive traffic from numerous sources, which results in service outages, financial losses, and reputational harm [171]. According to Nexusguard INT (2022), the frequency of DDoS incidents surged by approximately 75.60% in the first half of 2022 when compared to the latter half of 2021 [172]. Amazon reported a record-breaking DDoS attack in Q1 2020 that reached a rate of 2.3 Tbps [173]. A Spamhaus Botnet Threat Update in early 2022 noted the highest botnet activity originated in China with over 590,000 bots, followed by the USA with 376,000 and India with 350,000 [174]. As per Carero (2022), most DDoS incidents last less than ten minutes, yet during these brief intervals, enterprise organizations face losses ranging from \$10,000 to \$50,000 due to system downtime and mitigation efforts.

The frequency and sophistication of Denial-of-Service attacks are growing rapidly, causing considerable economic damage, particularly in the financial sector. These attacks flood networks with traffic, causing outages and interrupting essential services. A single DDoS incident can inflict direct and indirect losses worth hundreds of thousands of dollars per hour. One instance saw a 21-minute attack on a bank resulting in \$6,710 in direct financial loss due to 573,478 failed transactions [184]. This estimate excludes ancillary costs such as infrastructure repairs, customer dissatisfaction, and long-term reputational setbacks. To counter these threats, companies and ISPs employ protective technologies like Radware DefensePro, Cloudflare DDoS Mitigation, Akamai Edge DNS, and Azure DDoS Protection, which assist in identifying, reporting, and analyzing threats at multiple layers. However, the increasing speed, complexity, and distribution of modern attacks challenge the effectiveness of traditional scrubbing centers. Online systems often have unique identifiers, aligned with organizational goals for confidentiality, integrity, and availability. Based on threat models, it's evident that attackers primarily aim to compromise service availability using DoS

strategies. DoS attacks hinder legitimate access by disrupting services through unauthorized interference, as noted by CISA (2019) [175].

DDoS attacks can be categorized based on their targeted vulnerabilities: volumetric attacks consume network bandwidth, protocol attacks exploit server resource limits, application-layer attacks target web apps, amplification attacks leverage protocol weaknesses to intensify traffic, and low-and-slow attacks use limited bandwidth to avoid detection. Among these, volumetric and resource-based attacks are the most disruptive due to their scale and ease of deployment, making them harder to detect and counter in real time. The proposed work focuses on these types and aims to deliver a comprehensive detection approach. Conventional statistical models often fail to capture the complexity of DDoS patterns and tend to generate high false positives, especially for subtle attacks [176]. Moreover, these legacy techniques are generally inadequate for real-time processing and handling the high throughput of modern networks. Deep learning models offer scalable and adaptable solutions, capable of learning complex attack patterns and classifying new threats without manual rule definitions. These models are able to work with big datasets and can find attack almost in real-time when using modern computer hardware [177]. They can also be updated while working which helps them change with new attack behaviors. But even with these advantages, there are still some problems. They do not always adapt well to new types of features, and they often do not use methods that learn from groups of data samples, like multi-instance learning.

A new model called Multi-Branched Hybrid Perceptron Network (MHHPN) is introduced. This model puts together different advanced methods in one system. MHHPN has three different parts (branches): the first is a traditional Multi-Layer Perceptron (MLP) that helps to get broad and general features from the data, the second is based on DenseNet and helps to reuse features and make learning easier and faster, and the third is based on ResNet, which helps to stop problems with losing information during deep learning. Each of these branches has a special job in learning different kinds of DDoS attack features. The MLP part finds high-level patterns in the data, the DenseNet part helps to learn deep and complex features by linking many layers together, and the ResNet part helps the model remember deep information without losing it as the data goes through many layers.

After each part finishes its work, all the features are joined together into one combined feature vector. This helps the model understand the attack from many points of view. This combination makes the model better at finding different types of DDoS attacks, especially those that are large in size or try to use up system resources. At the end, the joined features go through fully connected (dense) layers to give the final result. This helps the model make accurate and strong decisions about whether the data shows normal or attack behavior. To make the model even more flexible, MHPN uses something called Dynamic Feature Adaptation. Model can keep changing and learning from new types of attacks. It includes a Multi-Instance Learning (MIL) method also. This lets the model learn not just from single data points, but from groups of related traffic data. This helps the model find and understand more advanced or hidden attacks. MHPN gives better results than traditional models in terms of accuracy and how well it handles different situations when compared using different datasets. The attention mechanism inside the model helps to focus on the most important features, and the special DenseNet and ResNet connections help the model learn better and faster without problems. These features collectively make MHPN suitable for tackling the challenges of modern DDoS detection.

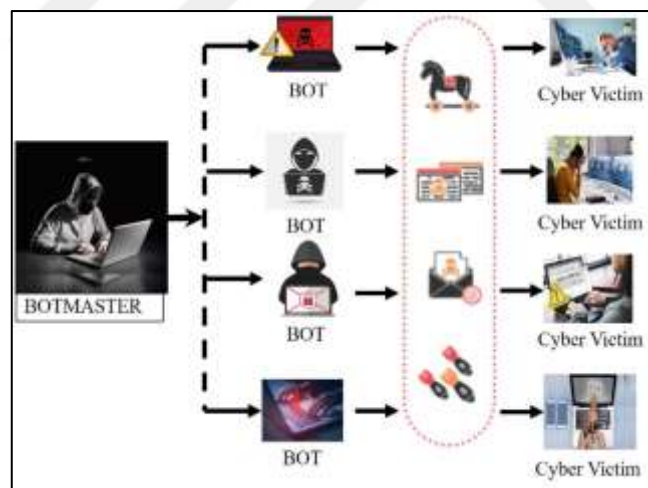


Figure 4.1: Working with Botnets to Execute A DDoS Attack.

DoS attacks have become increasingly frequent and have become sophisticated. It has caused heavy business losses, and the banking sector is the most attacked. They tend to overwhelm networks with traffic, rendering services unavailable and causing essential operational disruptions.

One DDoS attack can cost hundreds of thousands of dollars per hour regarding direct loss in revenue and indirect cost in service downtimes. For instance, a 21-minute DDoS against the bank in one attack led to financial losses of about \$6,710, caused by a total of failed transactions amounting to 573,478 during the downtime [184]. This figure only accounts for direct transaction income loss, while the broader financial impact extends to repair costs, infrastructure upgrades, customer dissatisfaction, and long-term reputation damage. To avoid these threats, ISPs and companies use solutions and scrubbing centers like Radware DefensePro, Cloudflare DDoS Mitigation Services, Akamai Edge DNS, and Azure DDoS Protection [171]. Such solutions provide multiple levels of protection during threat identification, reporting, and analysis. However, modern DDoS attacks are getting faster, more complex, and more distributed, thus rendering the traditional scrubbing centers ineffective. Internet-based services can have different system IDs needed for the specific organizational requirements of confidentiality, integrity, and availability. Based on the attacker model, it is evident that attackers primarily aim for service availability where they use DoS attacks. A DoS attack is one in which a client cannot gain access to the systems and the resources he requires due to the unlawful interferences of the attackers CISA (2019) [175]. DDoS attacks can be divided into different types, and every attack aims to exploit specific network weaknesses. The main categories are volumetric ones, which attack the available network bandwidth; protocol ones, which target servant's resources; application layer ones that target resources at the application level; amplification ones that take advantage of protocols to multiply incoming traffic and low-and-slow ones that use restricted bandwidth but target weakness. However, Volumetric and resource consumption DDoS attacks are more disruptive as they can overwhelm network and server resources on a large scale, making them harder to detect and mitigate quickly. Their ease of execution makes them more prevalent than application layer attacks. Therefore, our study emphasizes both types of attacks and seeks to offer a promising solution. It has been discovered that simple statistical methods are inadequate in describing various attack characteristics, and many of the systems exhibit high false positive rates, especially for low-intensity attacks [176]. Further, traditional methods generally do not have the advantage of real-time processing required for high-speed networks. Thus, they are inadequate in dealing with large volumes or high-speed modern DDoS attacks threatening an organization's server.

The methods based on deep learning provide practical solutions to these challenges. These advanced techniques are thus very flexible, allowing them to learn patterns from the data and classify new attack types without being hardcoded. Deep learning models can handle large volumes of data and offer near real-time responses due to the availability of appropriate hardware [177]. Moreover, these models can learn continuously; they can update these models on the fly based on changing state and attack patterns. However, these approaches also have some limitations, such as a lack of dynamic feature adaptation, which hinders their ability to monitor the varying attack patterns, and the absence of a multi-instance learning. Realizing the abovementioned challenges, a novel Multi-Branched Hybrid Perceptron Network (MHHPN) framework addresses these limitations by integrating multiple advanced techniques into a unified architecture. MHHPN features a multi-branched structure combining three distinct approaches: a standard MLP for primary feature extraction, a DenseNet-like branch for efficient gradient flow and feature reuse, and a ResNet-inspired branch to mitigate the vanishing gradient problem. The motivation for using three different types of branches, MLP, DenseNet-like, and ResNet-like, is to capture a broad range of feature representations that will obtain effective DDoS detection. As the MLP branch focuses on extracting broad, high-level global patterns, the DenseNet-like branch learns deep, hierarchical features through feature reuse across layers; in contrast, the ResNet-like branch captures complex, long-term dependencies with residual connections, making deeper learning possible without information loss. The outputs of these branches are concatenated to form a comprehensive feature vector, enhancing the model's ability to capture diverse and intricate patterns associated with various types of DDoS attacks, including both volumetric and resource consumption attacks. This fusion scheme leverages the strengths of each branch, resulting in a robust and efficient detection mechanism. Our approach proposes a new fusion scheme where the combined features extracted from all the branches are subjected to fully connected layers, thereby giving out the final classification output. It keeps the modeling process internal to each branch, and thus, the features that it will ultimately have been diverse and robust.

To further improve detection efficiency, the MHHPN model includes Dynamic Feature Adaptation, enabling the network to modify its detection features corresponding to the dynamic aspect of attack methods. Second, the model works in the Multi-Instance Learning (MIL) scheme, suitable for dealing with network traffic data. In MIL, the model learns the

specific patterns within each bag and thus becomes better placed to detect multiple and different attack types. However, extensive experiments carried out in the proposed work on various attack datasets showed that MHHPN has higher accuracy and better robustness than the existing models and other developed hybrid architectures of MLP. The attention mechanism effectively helps to select essential features for the model, and the dense and residual connections improve the gradient flow and the propagation of features and techniques that address all the issues related to DDoS attacks.

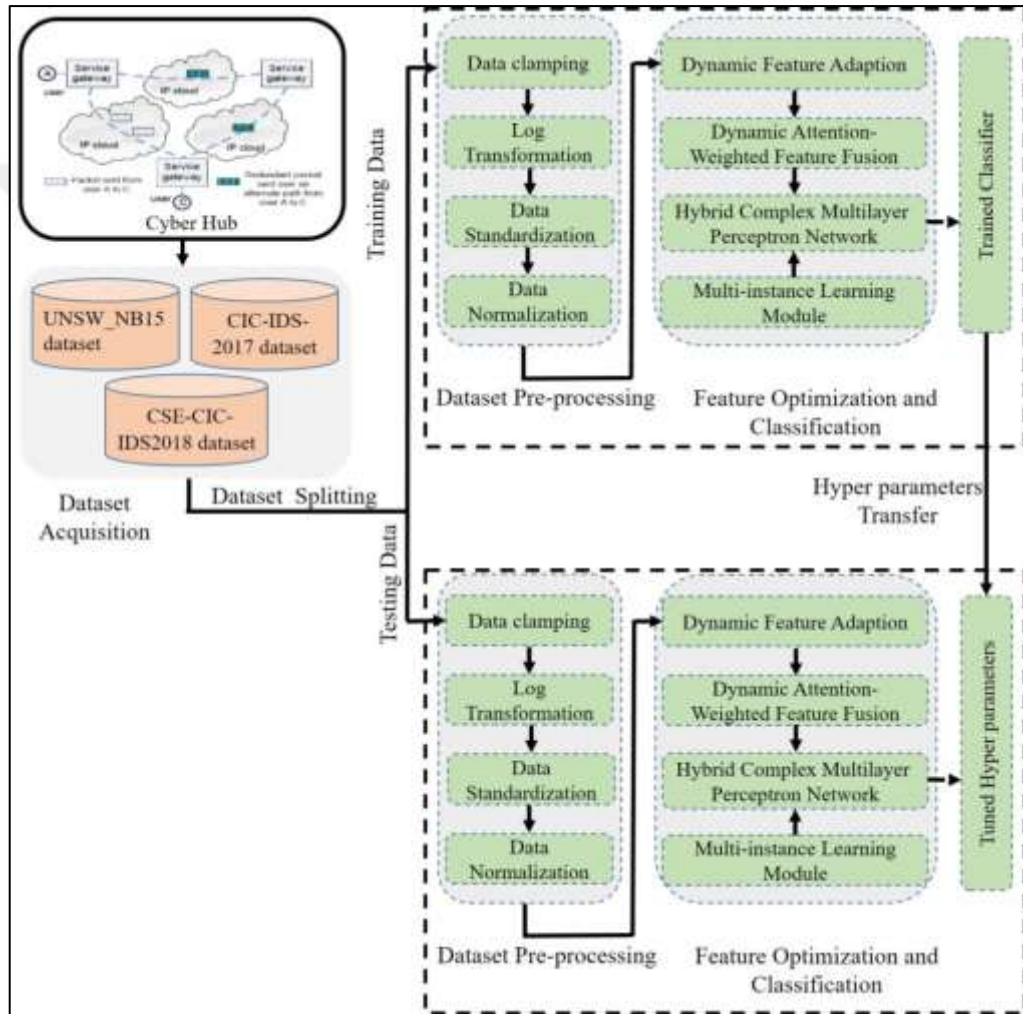


Figure 4.2: Pipeline Architecture of The Proposed Deep Learning-Based IDS Model.

The proposed work can be summarized into the following points: 1. An innovative architecture is introduced and named Multi-Branched Hybrid Perceptron Network (MHHPN) that involves three branches: MLP, DenseNet-like, and ResNet-like. Each branch processes input data differently, allowing the model to capture a broader range of features.

The working pipeline of the proposed methodology is shown in Figure 4.2. This integration leads to robust performance against various state-of-the-art methods, achieving 96.02% accuracy on UNSW-NB15, 99.99% on CIC-IDS 2017, and 99.96% on CIC-IDS 2018, all with perfect precision, recall, and F1 scores. 2. A novel Dynamic Feature Adaptation mechanism that helps the model learn new features in real-time as per the new emerging patterns of DDoS attacks. It increases the robustness of the model and its capability to identify new attack methods, which is still a weakness of standard CNNs. 3. Introducing an effective feature merging process, namely "Dynamic Attention-Weighted Feature Fusion," to amalgamate attributes generated by MLP, Dense, and ResNet branches. 4. MIL for the hybrid architecture has been used, where the sets of instances (bags) are used to train the model, which enhances its capability to handle the nature of the network traffic data. 5. The MHHPN model is rigorously evaluated using five-fold cross-validation on three publicly accessible datasets: When compared with traditional models and other state-of-the-art hybrid architectures, the suggested architecture outperforms CIC-IDS-2017 [179], UNSW-NB15 [180], and CSE-CIC-IDS2018 [181] concerning accuracy and robustness.

4.2 DATASETS FOR INTRUSION DETECTION

To thoroughly assess the accuracy and adaptability of the proposed Multi-Branched Hybrid Perceptron Network (MHHPN) combined with Multi-Instance Learning for identifying Distributed Denial of Service (DDoS) attacks, the proposed work employs three widely accepted and publicly accessible benchmark datasets: UNSW_NB15, CIC-IDS-2017, and CSE-CIC-IDS2018. These datasets are well-regarded for their ability to emulate both legitimate and malicious network activities, making them highly suitable for the development and validation of intrusion detection systems in practical environments. Every dataset subsequently provides a different combination of attack types, network traffic patterns, and system settings. These variations are utilized to verify the intrusion detection methods in an extensive and unbiased manner. They also guarantee that the model is not only optimal for one type of data but will perform well across different conditions and setups.

4.2.1 UNSW_NB15 Dataset

The network traffic is sniffed using Tcpdump, a popular packet analysis tool, and about 100 GB raw PCAP data is generated. The dataset is designed to mimic real internet traffic, and

normal behavior along with different types of attacks is included in the dataset. The dataset is thus very helpful for researchers and developers who are working on Intrusion Detection Systems (IDS).

There are nine different forms of attack within this data set, and each illustrates a unique way that systems can be hacked. There are nine different types of cyber-attacks within this data set, and each illustrates a unique way that hackers might try to break or destroy computer systems. Fuzzers are attacks that send strange or tainted data to a machine to see if it crashes or reacts inappropriately. Analysis attacks are scanning for open ports or looking at what programs a machine has installed in an attempt to find out where a system would be vulnerable. Backdoors are secret ways to get into a system without needing the usual password or login. Denial of Service (DoS) attacks try to stop a system from working by sending too many requests or using up its resources. Exploits use known problems or bugs in software to attack the system. Generic attacks include brute-force methods where a hacker tries many passwords until the right one is found. Reconnaissance is when an attacker collects information about a system or network to help plan other attacks. Shellcode attacks happen when bad code is added to a program and run. Worms are harmful programs that can copy themselves and spread from one computer to another across a network. Each of these attacks shows how dangerous and smart cyber threats can be and why systems need strong protection.

To get useful details from the raw data, tools like Argus and Bro-IDS (now called Zeek) are used. A total of 12 algorithms are applied to get 49 features from the traffic data. These features include basic information such as how long a packet lasted, the source and destination IP addresses and ports, and the number of bytes. It has more advanced information like protocol flags, direction of traffic, and TCP state. All of these along with appropriate labels for each traffic record (normal or attack) are stored in a file named UNSW-NB15_features.csv, which is included along with the dataset. The UNSW-NB15 dataset is very robust and complete as it is a mix of real traffic as well as different types of attacks. This is well-suited specifically for training and testing intelligent intrusion detection models, including the proposed Multi-Branched Hybrid Perceptron Network (MHHPN). The heterogeneity and depth are good for constructing models with adequate capacity to generalize nicely in real-world cyber security scenarios.

4.2.2 CICIDS2017 Dataset

This dataset solves some problems in older datasets like KDD'99 and NSL-KDD. It includes more realistic and newer kinds of cyber-attacks, so it is more appropriate to test and train security systems. It also includes a more balanced ratio of normal and attack traffic, so machine learning models can learn more fairly. This is discussed more fully in Chapter 3, Section 3.2.2.

4.4.3 CIC-DDoS2019 Dataset

The CSE-CIC-IDS2018 dataset is a new and worthy dataset in cyber security research. It is an upgrade to the earlier work on the CIC-IDS-2017 dataset [13]. The dataset was created in partnership with the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). The main objective was to create a realistic simulation of a network of a large business. It was crafted to look like the kind of networks that appear in real enterprises today, which makes it extremely appropriate for use in assessing the performance of intrusion detection systems (IDS). Seven different types of attack scenarios are present in the dataset:

- i. Brute Force: Attempts to breach login interfaces by password guessing
- ii. Heartbleed: Takes advantage of memory disclosure vulnerabilities, Botnet, synchronized attacks from various compromised devices and DDoS: Disruption of services through traffic flooding,
- iii. Web Attacks: Exploits aimed at web server vulnerabilities,
- iv. Infiltration: Malware that spreads from external networks into internal systems,
- v. Network Infiltration: Attacks originating from within the network, targeting critical infrastructure.

To effectively evaluate real world network environment the CSE-CIC-IDS2018 dataset was created using a complex and realistic network setup. It includes 50 attacker machines that try to break into a simulated company made up of 420 client computers and 30 servers. These servers are organized into five departments with different roles, such as finance, human resources (HR), and research and development (R&D). The cyber-attacks were carefully

designed to look like normal network traffic, which makes it harder for intrusion detection systems (IDS) to tell the difference between safe and harmful behavior. During the test, all the network traffic and system logs from each machine were collected. This data was then processed using a tool called CICFlowMeter-V3, which pulled out up to 80 different features from each flow of network data. These features include things like how long a flow lasts, how the size of packets changes, the length of headers, and information from higher levels like which HTTP methods were used and how long the web addresses (URIs) were. The strength of the CSE-CIC-IDS2018 dataset is in how real it feels it uses a company-like setup, has many kinds of attacks, and includes both attacks from inside the network and from outside. This makes it very helpful for testing IDS models, especially to see how they handle smart and sneaky attacks, and how well they adapt to new types of threats.

The UNSW-NB15, CIC-IDS-2017, and CSE-CIC-IDS2018 datasets were chosen because they each have a wide range of features, cover many different types of cyber-attacks and do a good job of simulating real-world network activity. These data sets together create a strong test bed for the above-discussed MHHPN model to detect a broad range of attacks like Distributed Denial of Service (DDoS), covert intrusions, and botnet behavior. Another strong motivation behind the choice of these data sets is that all of them are comprised of labeled traffic, needed for machine learning and hybrid model training and testing. Both the techniques are core elements of the MHHPN system. By using over one dataset, the model can be tested more fairly and avoid becoming overly specialized in one kind of data. This helps to improve the reliability and accuracy of the model when used in real-life situations. Overall, joint utilization of UNSW-NB15, CIC-IDS-2017, and CSE-CIC-IDS2018 datasets provides a comprehensive and complete validation of the performance of the proposed model in identifying DDoS attacks and other forms of cyber-attacks in real-world networks.

Table 4.1: Data Partitioning of Proposed Model on Datasets.

Dataset	Label	Training set	Validation set
UNSW_NB15 dataset	Normal	175,341	82332
	Attack		
CIC-IDS-2017 dataset	Normal	248073	62018
	Attack		
CSE-CIC-IDS2018 dataset	Normal	2,117,553	529388
	Attack		

4.3 DATA PRE-PROCESSING TECHNIQUES

Data preprocessing was initiated by choosing 73 features from three most commonly employed datasets, namely UNSW-NB15, CIC-IDS 2017, and CIC-IDS 2018. Features were chosen based on high-density only with more than 80% non-zero values and being meaningful in the direction of DDoS attack detection, i.e., packet rate, flow duration, mean packet size, and connection state. These are the most notable attributes of network traffic. They allow for effective differentiation between normal and anomalous activity that makes them useful for detecting volumetric and resource consumption-based attacks. Data quality is improved with several pre-processing techniques. The following are the details of the specific pre-processing methods employed, including data clamping, log transformation, data standardization, and data normalization, each designed to address different aspects of data variability and ensure more robust and accurate model training.

i. Data clamping

The clamping or Winsorization procedure [185] is applied to the gathered data to prevent the presence of outliers within the data, which leads to unbalancing and affects the models' performance. This method replaces values that fall above or below specified percentiles, replacing them with the respective percentile values.

For a given feature vector: $X = \{X_1, X_2, \dots, X_n\}$, let $\max(x)$ be the maximum value of the feature, $\text{median}(x)$ be the median value of the feature and $P_{95}(x)$ be the 95th percentile value of the feature

The implemented clamping process can be defined using the following conditions:

Condition 1: $\max(x) > 10 * \text{median}(x)$

Condition 2: $\max(x) > 10$

Clamping Rule: If both "Check Conditions" are true, then each element x_i in the feature vector will be represented according to the following rule in Eq.4.1:

$$x'_i = \begin{cases} x_i & \text{if } x_i \leq P_{95}(x) \\ P_{95}(x) & \text{if } x_i > P_{95}(x) \end{cases} \quad (4.1)$$

Therefore, any value x_i in the feature vector that exceeds the 95th percentile value $P_{95}(x)$ will be clamped to $P_{95}(x)$.

ii. Log Transformation

Log transformation is a standard mathematical technique that reduces skewness, scales up significant differences, and decreases variance [186]. It redirects multiplicative relationships to additive forms, which makes the data cleaner for statistical analysis. This process minimizes the effect of such extreme values, making machine learning models perform better by improving the spread of the data. The formula of Log Transformation is defined in Eq. 4.2.

$$y_i = \log_2(x_i + 1) \quad (4.2)$$

This formula makes all the values positive and scales down the larger values, moving them closer to the sizes of the smaller values, thus improving the removal of skewed data and skewness.

iii. Data Standardization

Data standardization [187] is a crucial step in the pre-processing of data where all the scales of the numerical features have 0 as the mean value and features have a standard deviation of 1. This process scales the features, thus making them more comparable, and it ensures that features with large scales do not tend to dominate the model training. In our case, standardization is revealed as a procedure that significantly decreased the range of data, especially values at the right tail of the distribution. This compression minimizes the impact of highly influential values outside, thus improving the performance level of computers when carrying out the learning process. Therefore, normalizing allows for a better distribution of the given data set and more concisely optimized training. The formula for data standardization is shown in Eq. 4.3.

$$z = \frac{x - \mu}{\sigma} \quad (4.3)$$

where x is the original feature value, μ is the feature's mean, σ is the feature's standard deviation, and z is the resulting standardized value.

iv. Data Normalization

Normalization [188] is one of the data pre-processing steps that scale features to the standard deviation for simplicity; often, it is done between 0 and 1. It enables better performance and stability of various machine learning algorithms. Proposed work employed a min-max normalization approach to scale all the standardized variables using formula 4.4.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4.4)$$

In the data pre-processing phase, relevant features are selected from multiple datasets based on their significance to the task. Several techniques are applied to enhance data quality and model performance. Data clamping is used to handle outliers by adjusting values beyond a certain percentile. Log transformation reduced skewness and normalized the data by scaling larger values closer to smaller ones. Data standardization standardized features to have a mean of 0 and a standard deviation of 1, ensuring comparability. Lastly, data normalization scaled values to a defined range, improving the stability and performance of machine learning models.

4.4 DYNAMIC FEATURE ADAPTATION MECHANISM

Dynamic Feature Adaptation [178] is one of the most essential components for improving the performance of deep learning models by highlighting the critical features in each context of the input data. Such flexibility enables the model to pay more attention to essential features of the values, which enhances the model's capability to identify intricate trends and interactions. At this moment, it overcomes some shortcomings in practice, such as feature redundancy and noise, which significantly influence the reliability of predictions. The versatility of the feature representation in situations like the detection of a DDoS attack is essential for such a process since the data variety and richness require a stable and adaptable representation.

Here, the Dynamic Feature Adaptation applied to the individual deep features extracted by Perceptron, ResNet, and DenseNet branches helps fine-tune and optimize each branch's dynamic features according to their importance and relevance. Each branch processes the input data and generates a feature vector, denoted as F_{B_i} where $B_i = \{\text{Perceptron, DenseNet, and ResNet}\}$ for $i \in 1, 2, \text{ and } 3$. Let $F_{B_i} \in \mathbb{R}^{B_i}$ be the feature vector with B_i dimensions. Further, a dynamic weighting mechanism is applied by assigning different weights to the

individual features in F_{B_i} based on their relevance or importance. Here, the dynamic weight vector is computed according to Eq. 4.5.

$$W_P = [W_1, W_2, \dots, W_{B_i}] \quad (4.5)$$

Where weight W_i refers to the weight for i^{th} feature in the feature vector F_{B_i} . These weights are determined using Eq. 4.6.

$$W_i = \text{softmax} (W_{att} * F_i + b_{att}) \quad (4.6)$$

Here, W_{att} and b_{att} are learnable parameters of the attention mechanism and the Softmax function confirms that the weights sum to 1, providing a probabilistic interpretation. The calculated dynamic weights are applied to the feature F_{B_i} to design an adapted feature vector $F_{B_i}^{adapted}$ according to Eq. 4.7.

$$F_{B_i}^{adapted} = W_P \odot F_{B_i} \quad (4.7)$$

Here, $\odot$ denotes element-wise multiplication. Each feature in F_{B_i} is scaled by its corresponding weight in W_P . Finally, the adapted feature vector $F_{B_i}^{adapted}$ is passed through additional fully connected layers to further refine the features using below Eq. 4.8.

$$\text{Refine} (F_{B_i}^{adapted}) = \sigma (W_{refine} * F_{B_i}^{adapted} + b_{refine}) \quad (4.8)$$

Where, W_{refine} and b_{refine} are weights and bias parameters of the refinement layer, and σ is the ReLU activation function. This approach addresses feature redundancy and noise in DDoS classification through several key strategies: it computes dynamic weights through an attention mechanism that makes it focus more on relevant features, down-weighting or ignoring redundant ones. Figure 4.3, shows the steps of the Pseudocode for Dynamic Feature Adaptation in Figure 4.4.

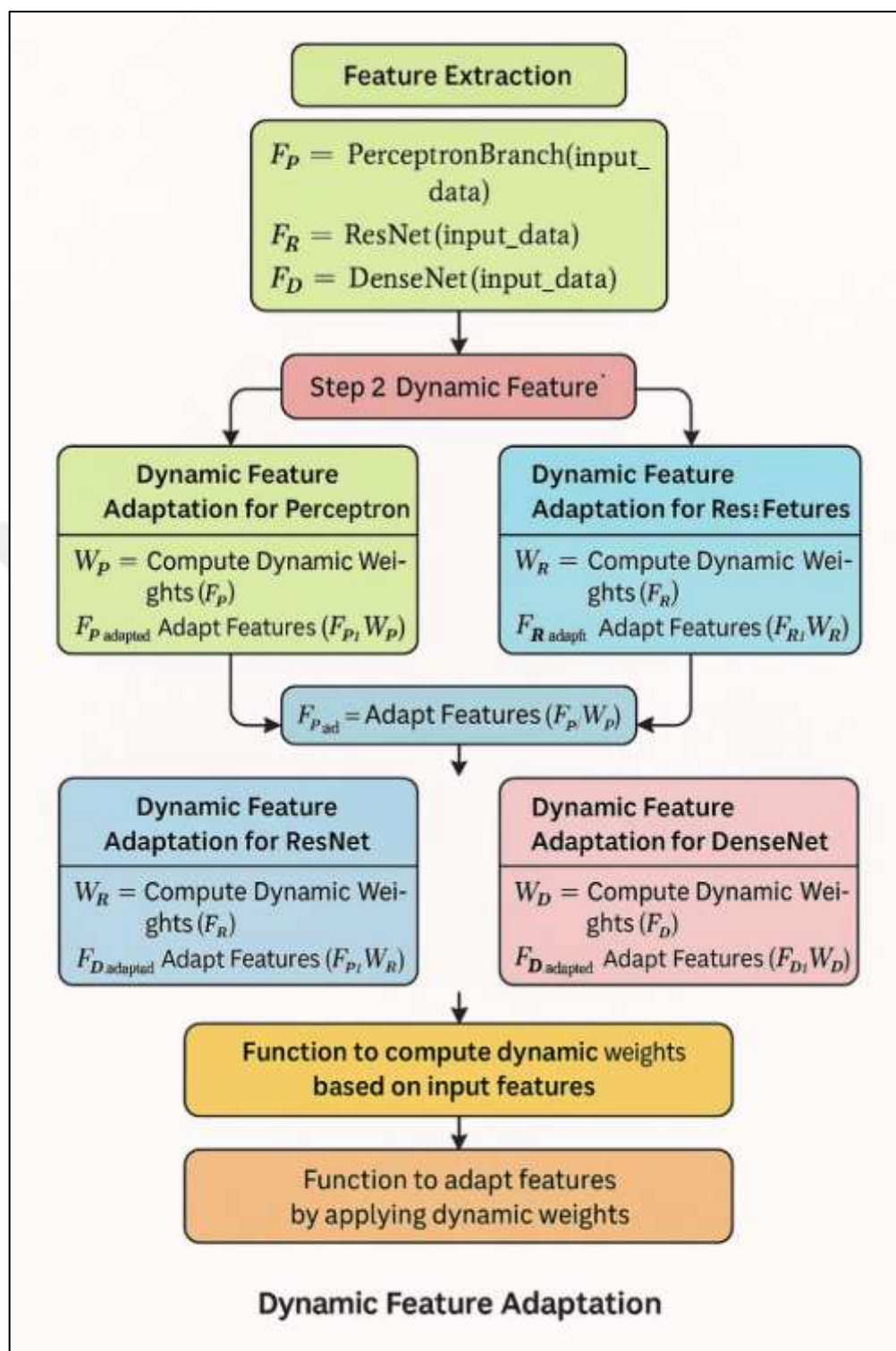


Figure 4.3: Shows the Steps of The Pseudocode for Dynamic Feature Adaptation.

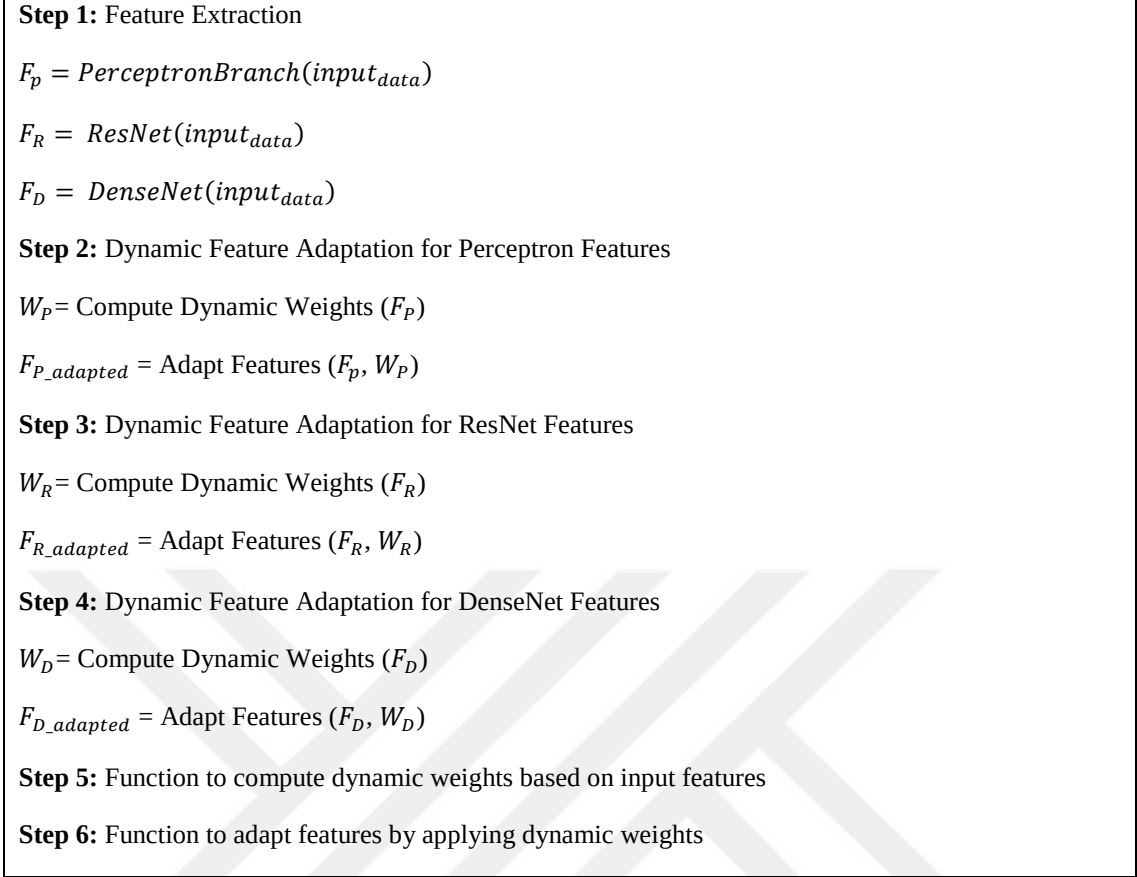


Figure 4.4: Illustrates Pseudocode for Dynamic Feature Adaptation.

The applied attention mechanism mainly focuses on features that recognize abnormal traffic patterns and resource strains, focusing on high-impact indicators such as traffic rate metrics and connection states. This selective emphasis enhances the model's responsiveness to crucial markers of both volumetric and resource-based DDoS attacks, thereby improving its detection capacity across varied attack types. In this way, it would be the adaptive selection of which features will be emphasized while cutting down on the noise and redundancy of the feature set. The element wise feature multiplication by the weights of its corresponding feature component will have fine control over how much each feature can contribute in a very subtle way. Low-priority features will be given low weights, while their effects are negligible in the output. The flexibility is useful in real-time environments, especially when the DDoS attack pattern changes; it filters noise and adapts to data patterns that change in real-time. The pseudo-code of the proposed dynamic feature adaption is shown in Algorithm 4.1.

4.5 MULTI-BRANCHED HYBRID PERCEPTRON NETWORK

The architecture of the hybrid complex multilayer perceptron (MHHPN) network involves integrating three distinct branches: A standard Multilayer Perceptron (MLP) branch, the DenseNet-like branch, and the ResNet-like branch. All branches operate on the input data differently, which advances the feature extraction process. The architecture of the proposed MHHPN network is shown in Figure 4.5.

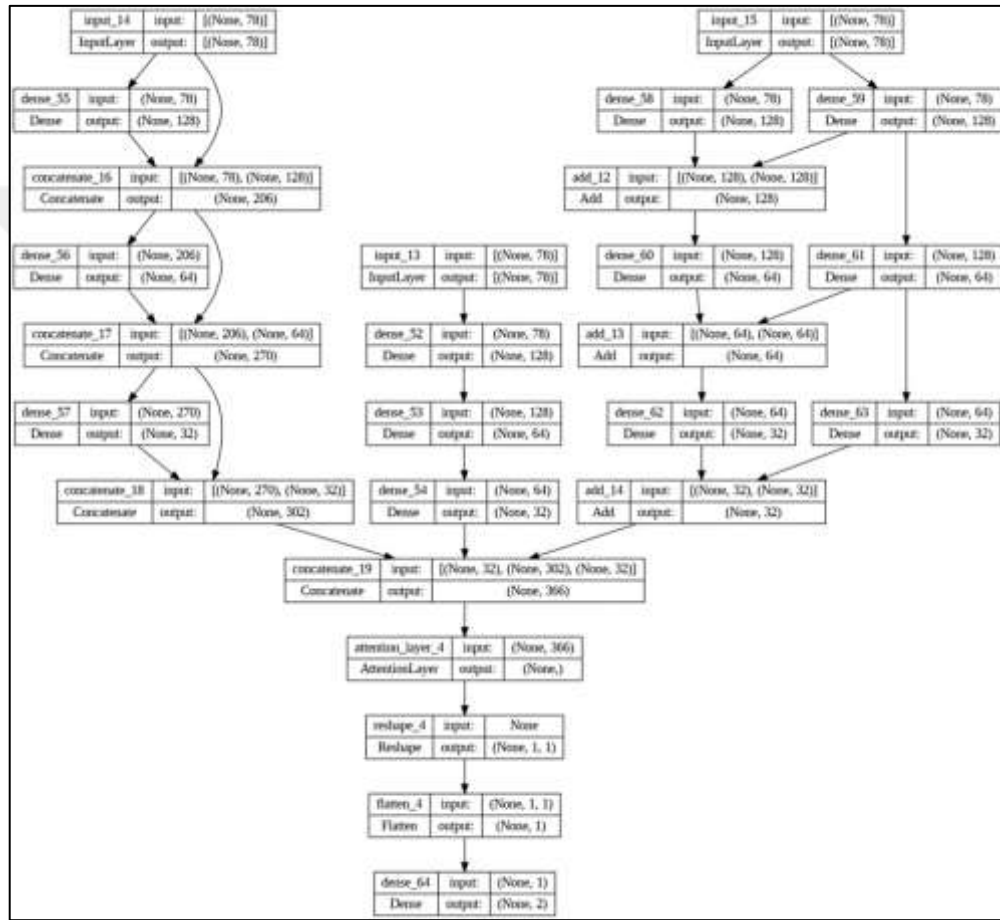


Figure 4.5: The Architecture of The Proposed Hybrid Complex Multilayer Perceptron (MHHPN) Network.

The basic structure of the MLP branch is simple, and it contains three fully connected layers with the neuron number decreasing gradually (128, 64, and 32), and the ReLU activation function is used. This branch obtains simple kinship information from the input data, such as the basic hierarchy. The MLP has been organized so that neuron counts are reduced step by step to lower the dimensional space but in a way that important information is retained.

This organization captures more superficial patterns of kinship or relationship within the data by successively abstracting away secondary features while concentrating on primary characteristics. As the number of neurons in an MLP is reduced from one layer to the next, it compresses data into essential global features while ignoring noise details. The process is relatively straightforward and guarantees that vital attributes are identified without introducing many computations. This step can be calculated as in Eq. 4.9.

$$F_{MLP} = Dense_{Relu} (128, Dense_{Relu} (64, Dense_{Relu} (32, X))) \quad (4.9)$$

Where X is the original feature vector. The DenseNet-like branch is also initiated with an FC layer of 128 neurons, using concatenations of previous layer outputs as the Dense block. This branch has several cardinally packed layers with concatenations intended to advance feature reclusion and improve the gradient flow that boosts the learning. The DenseNet-like branch, which concatenates the feature maps of the previous layers, is supposed to learn detailed patterns and dependencies in the data, making it more useful for feature extraction. This step can be calculated as given in Eq. 4.10.

$$F_{DenseNet} Concat ([X, Dense_{Relu} (128, Concat ([X, Dense_{Relu} (64, Concat ([X, Dense_{Relu} (32, X)])))])) \quad (4.10)$$

Finally, the primary ResNet-like branch incorporates skip connection ideas, allowing the network to learn residual mappings. Beginning with the list of parameters with a fully connected layer of 128 neurons, it also includes additions of input transformations before each dense layer (128, 64, and 32 neurons). These skip connections also tackle the vanishing gradient problem and training rate enhancement. This step can be calculated as in Eq. 4.11.

$$F_{ResNet} Add([X, Dense_{Relu} (128, Add ([X, Dense_{Relu} (64, Add ([X, Dense_{Relu} (32, X)])))])) \quad (4.11)$$

4.6 FEATURE FUSION BY DYNAMIC ATTENTION-WEIGHTED FEATURE FUSION

After extracting adaptive feature representations from these three branches, a dynamic attention-weighted feature fusion function is introduced that fuses these features semantically meaningfully [61]. Here, the mechanism encompasses multi-head attention,

self-attention, and gating mechanisms, so the features' importance determines the correct weighting and computing procedure. In our proposed fusion mechanism, for each adaptive feature set F_i (where i represents MLP, DenseNet, or ResNet), a multi-head attention is applied to capture different aspects of the features as shown in Eq. 4.12.

$$\text{MultiHead}(F_i) = \text{Concat}(\text{head1}(F_i), \text{head2}(F_i), \dots, \text{headH}(F_i)) \quad (4.12)$$

where H is the number of attention heads. Self-attention scores for each feature branch using a self-attention mechanism are defined in Eq. 4.13

$$\text{SelfAttention}(F_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) \quad (4.13)$$

where Q_i, K_i and d_k are the query, key, and value matrices derived from F_i , and d_k is the dimension of the key vectors. This mechanism helps in weighing the input features based on their relevance to each other. Further, a gating mechanism is used to enhance the contribution of each feature branch. The gate for each feature F_i is computed as in Eq. 4.14:

$$G_i = \sigma(W_g F_i + b_g) \quad (4.14)$$

where σ is the sigmoid function, and W_g and b_g are learnable parameters. The gating mechanism dynamically adjusts the influence of each feature based on its importance. During this gating mechanism, the dynamic attention weights are calculated using a softmax function to ensure that they sum up to 1. This can be calculated using Eq. 4.15

$$\alpha_i = \frac{\exp(G_i)}{\sum_{i=1}^3 \exp(G_i)} \quad (4.15)$$

Gating mechanism is used to enhance the contribution of each feature branch. The gate for each feature F_i is computed as in Eq. 4.16.

$$G_i = \sigma(W_g F_i + b_g) \quad (4.16)$$

where σ is the sigmoid function, and W_g and b_g are learnable parameters. The gating mechanism dynamically adjusts the influence of each feature based on its importance. During this gating mechanism, the dynamic attention weights are calculated using a softmax function to ensure that they sum up to 1 as shown in Eq. 4.17:

$$\alpha_i = \frac{\exp(G_i)}{\sum_{i=1}^3 \exp(G_i)} \quad (4.17)$$

where G_i represents the gated attention score for the feature F_i . This step ensures that the features are weighted appropriately before fusion. Finally, Eq. 4.18 represents features from different branches are fused using the dynamically calculated attention weights:

$$F_{Fused} = \alpha_1 F_{MLP} + \alpha_2 F_{DenseNet} + \alpha_3 F_{ResNet} \quad (4.18)$$

This final fused feature vector F_{Fused} combines the most relevant aspects of the input features, weighted by their dynamically adjusted importance. By leveraging the strengths of MLP, DenseNet, and ResNet branches and dynamically fusing their features using multi-head attention, self-attention, and gating mechanisms, our MHHPN network effectively captures and combines diverse feature representations, improving performance on various tasks. The pseudo-code of the fusion process is given in the Figure 4.6 and figure 4.7.

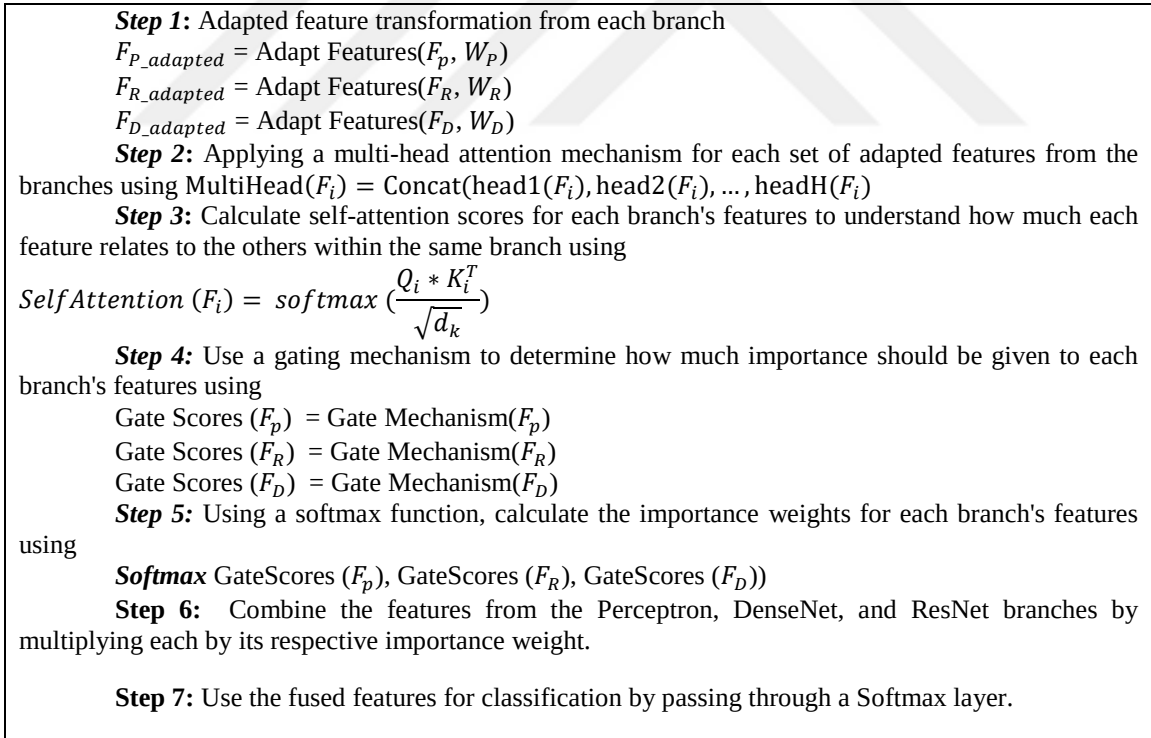


Figure 4.6: Illustrates Pseudocode for Feature Fusion Process by Dynamic Attention-Weighted Mechanism.

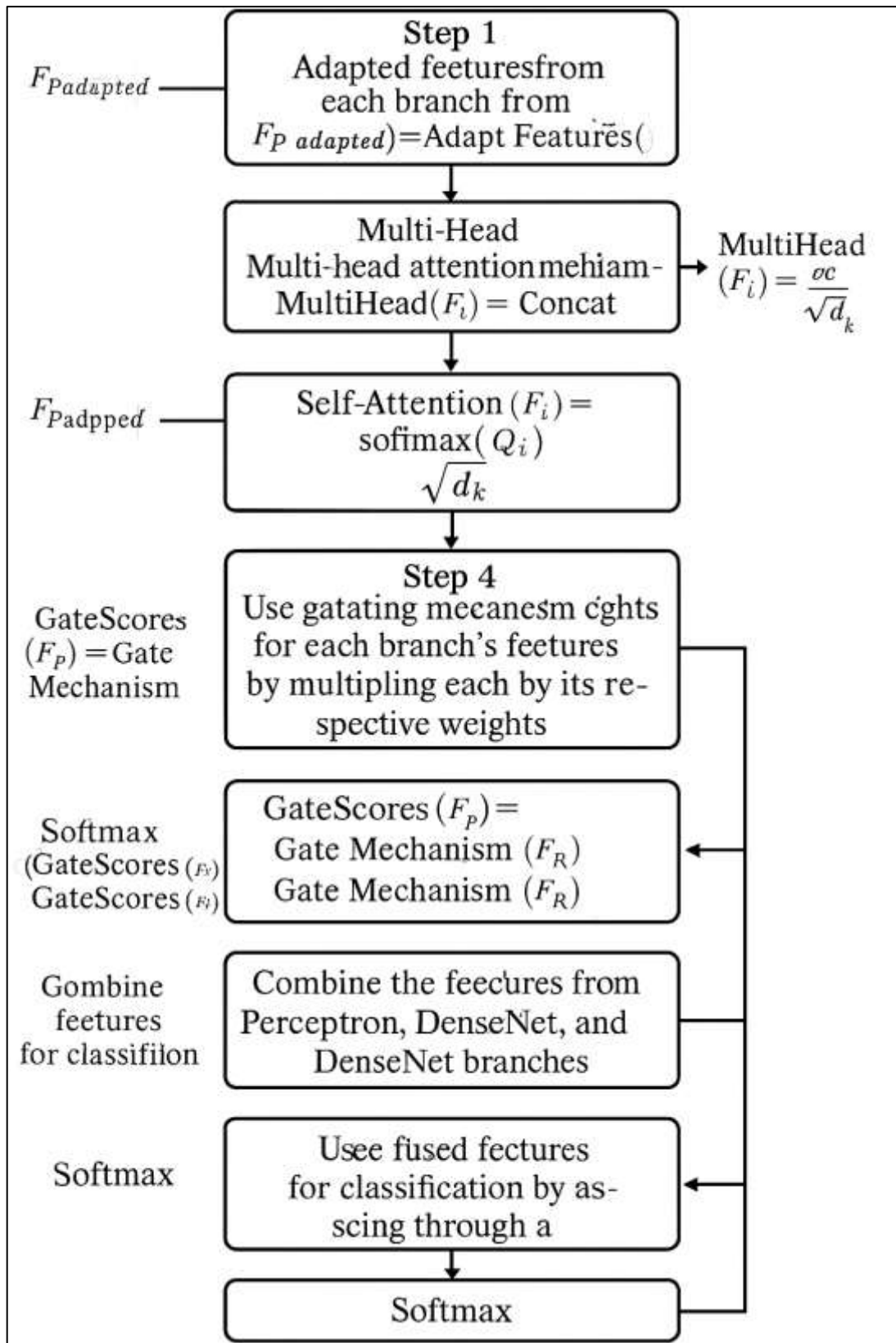


Figure 4.7: Pseudocode for Feature Fusion Process by Dynamic Attention-Weighted Mechanism.

4.7 EXPERIMENTAL SETUP

The experimental procedure for assessing the MHHPN network architecture is arranged in a manner that allows for establishing the best conditions for the model's performance and setting up replicability. The experiments are performed on a relatively robust hardware and software platform with components of modern computing technology. The hardware configuration is a server with two or more NVIDIA GPUs in each of them, each with no less than 12 GB of VRAM for parallel computations with large amounts of data. The server is installed with a multi-core processor, 64 GB of RAM, and a high-speed SSD disk for dealing with extensive data and shortening the time taken in training. Development environment is established in the Linux environment (Ubuntu 20.05 LTS). The environment is robust and can serve the machine learning environment. TensorFlow 2.8 is the most popular, default deep learning environment used during the model developing and training process of the MHHPN model. It has the capability to offer stability to humongous neural networks and accelerate the training process on the GPU device. Python libraries are used for pre-processing and augmentation of data. Used libraries are NumPy for numerical data processing, Pandas for structured data processing, and Scikit-learn for data normalization and data splitting. Training is made easy with the use of a sequence of TensorFlow's. Pre-defined callback functions as well as custom callback functions are used to track and enhance the training performance. Early stopping, model check pointing, and learning rate reduction are used to prevent over-fitting as well as model convergence.

4.8 INTEGRATION OF SDNETC WITH MHHPN-BASED DETECTION FRAMEWORK

Effective detection of Distributed Denial-of-Service (DDoS) attacks in modern networks not only needs robust classification models but intelligent and dynamic network infrastructure that can support real-time processing, flexibility, and scalability. The proposed Multi-Branched Hybrid Perceptron Network (MHHPN) has high performance in detecting DDoS attacks based on its dynamic feature adaptation and multi-instance learning capabilities. Bringing it to the full realization in a real, high-speed network scenario with an intelligent network infrastructure is essential. Cognitive Software-Defined Networking is designated as SDNetc. SDNetc operates data planes through the integration of AI-driven decision-making in the control plane. SDNetc provides a programmatic, centralized, and versatile architecture

for controlling complex network security tasks. It discusses how the central control and programmability in SDNetc enable efficient MHPN deployment, how dynamic redirection of real-time traffic during inspection is established, and how the design enhances both responsiveness and scalability in detecting DDoS attacks.

4.8.1 Centralized Control and Programmability in SDNetc for MHPN Deployment

Legacy networks are scattered and hard to reconfigure. This makes it challenging to leverage high-performance machine learning Intrusion Detection Systems like MHPN. SDNetc is different. It moves control logic to a central software controller. The controller possesses system-wide knowledge of the network and can make decisions for the whole system. Because of this, it is much easier to install MHPN models into the corresponding sections of the network.

The SDNetc controller can initiate and manage MHPN detection modules in different devices like switches, routers, or virtual machines. Rules can be initiated in the control layer when the network is completely programmable. This makes one change the network behavior based on the type of threat. A module of MHPN is called if the system detects traffic that looks like a volumetric attack. SDNetc can also cooperate with orchestration tools and AI systems that supervise the performance of the network and whether there exists any threat. These tools dynamically scale, retrain, or move the MHPN models based on changing traffic or emerging threats. Centralized management saves computer power and maintains the MHPN models at their maximum levels at all times.

4.8.2 Real-Time Traffic Redirection for Dynamic Analysis

One of the most stringent demands for the detection of DDoS attacks is real-time monitoring of the traffic. Modern networks handle extremely large amounts of high-rate data, and it is unrealistic to scrutinize each packet at each point. Instead, intelligent traffic diversion must be employed in order to focus detection efforts on the suspicious-looking flows. SDNetc is helped in this by its accurate flow control functionality. SDNetc continuously supervises traffic flows via flow statistics it collects from the data plane. The SDNetc controller is watching for signs of problems such as sudden traffic bursts, anomalous use of network protocols, or orphaned traffic from expected patterns. Whatever is found to be suspicious,

SDNetc can immediately redirect such traffic to MHHPN modules to be examined more thoroughly.

This smart diversion avoids adding additional burden to normal traffic. It saved processing capacity and only submits malicious data to a deep scan by MHHPN. It avoids spreading malicious traffic across the network by acting in a timely manner. Traffic redirecting policies can differ depending on the type of DDoS attack. For example, volumetric attacks may necessitate bandwidth inspection, but protocol attacks would necessitate packet header inspection. There are two kinds of analysis handled by the system. In more serious cases, the traffic can be redirected straight to MHHPN for immediate blocking. In less serious cases, a copy of the traffic can be redirected to be analyzed without disrupting the original flow. This kind of flexibility renders the detection system more responsive and resilient overall.

4.8.3 Enhanced Scalability and Real-Time Responsiveness

Scalability and quick response are highly important to any modern security system in the case of large and distributed DDoS attacks. It can affect thousands of devices at once. SDNetc helps solve these problems through its built-in ability to scale and manage things in real time. In SDNetc, new detection tools like MHHPN modules can be added across many network nodes without interrupting the network. MHHPN modules run as virtual network functions (VNFs) Network function virtualization (NFV). The SDNetc controller can give these modules the right amount of CPU, memory, or bandwidth based on how much the network needs at that moment. SDNetc is programmable so it can quickly share detection and blocking rules across the network. For example, if MHHPN finds a bad IP address or subnet then the SDNetc controller can update all the edge switches to block or slow down traffic from that source. The network reacts fast and gives strong protection by thereafter.

Recovery also works in case of fast response. Fixing a broken node in a network to restore protection can take a long time. But with SDNetc, the system can find out if a node fails and quickly send traffic to another MHHPN module. This keeps protection active with no delay.

4.8.4 Pattern of Attacks

One of the biggest challenges to intrusion detection is that computer attacks are continually evolving. Attackers continuously revise their methods to avoid signature-based detection or

take advantage of new, previously unknown vulnerabilities. Because of this, a DDoS detection system must be able to detect known and new types of attacks. SDNetc is also strong in this aspect since it is capable of dynamic policy update through data analysis and threat intelligence. SDNetc incorporates smart functions like reinforcement learning, anomaly detection, and log analysis into the control layer. These enable SDNetc to learn from experience and refine its detection rules with time.

For example, whenever MHPN discovers some new attack pattern that has never been seen before, then the controller can learn a general rule from that pattern. It can then modify its detection policy in such a way that it can identify similar attacks later on. Such policies can be shared with all the regions of the network in real time, and the system is ready for potential threats. SDNetc also returns detection results so that it can enhance the response in the future. MHPN detection results can be sent back to the controller so that the traffic rules can be changed, the detection thresholds can be changed, or the traffic priorities can be changed. This provides a feedback loop for the detection so that it will become smarter and faster in the future. Flexibility is even more important in cloud or multi-tenancy network environments. Various levels of traffic and risk can be present among various tenants. SDNetc can have specific rules for each tenant but will keep the entire network safe. It provides strong and tailored protection to everyone.

4.9 SUMMARY

The research work introduces a sophisticated neural network model for the improvement of cyber threat detection. Focus is on identifying Distributed Denial of Service (DDoS) attacks via in-depth network traffic analysis. The Multi-Branched Hybrid Perceptron Network (MBHPN) model is a multi-branched, hybrid model specifically tailored. It has the ability to identify complex and high-dimensional traffic patterns. One of the key aspects of MBHPN is dynamic learning capability, which enables it to cope with different network behaviors. This, in turn, enables it to extend its application from DDoS detection to other cyber security problems. Multi-branch nature is one of the greatest strengths of this framework. It simultaneously processes local and global data patterns. These parallel approaches help the model discover complementary outcomes which may be rejected in traditional single-path-based frameworks. It promotes improved learning performance, greater flexibility, and stronger resilience when handling real network traffic which in the majority of situations is

irregular and imbalanced. Considering DDoS detection as a primary usage scenario, the MBHPN's modularity scales to other security scenarios. For example, the framework model may be utilized for signature-based attacks like SQL injection or malware execution behavior patterns. This renders it extremely practical in its application within more sophisticated cyber security practices. In an effort to render the model useful for use in real-time and high-speed environments, various types of optimizations are explained in the study. Some of the examples are techniques like pruning and quantization that allow diminishing the computational weights of the model without compromising its accuracy. It also explores in creating learning approaches like federated learning, online learning, and domain-knowledge-aware feature adjustment tools that enhance the model's capacity to address new and moving threats, mainly in decentralized or limited environments. Another novel component of this activity is the introduction of explainable AI (XAI). Adding XAI components, though, the system can be made transparent when it makes a decision so that cyber security professionals can understand why a certain threat detection is being performed. This transparency can aid in better tuning of policies and targeted threat response effectively. The existing MHPN model is also enhanced when used in an SDNeta software-defined network system capable of offering centralized control with flexibility. SDNeta's real-time diversion of unwanted traffic to the detection engine does not impinge on network performance when optimizing the analysis of threats. Its scalability and flexibility render it a prime candidate to be integrated into the MHPN, creating a smart and robust cyber security solution capable of evolving and growing as fast as the ever-changing threat landscape. Finally, the chapter presents guidelines for future research in hardware acceleration using GPUs and TPUs for facilitating quicker data processing and real-time detection. Parallel computation makes this model implementable in high-volume or low-latency scenarios. In all, this research addresses not only current DDoS detection issues but also lays down the foundation for more scalable, explainable, and adaptive cyber security systems. The following chapter presents results and findings that are obtained by implementing the suggested framework.

5. RESULTS AND DISCUSSION

5.1 INTRODUCTION

In this chapter, a comprehensive performance analysis and comparative study of the proposed multi-branched hybrid architecture for intrusion detection is presented. Using datasets such as NSL-KDD, CICIDS2017, and UNSW-NB15, different measures such as Accuracy, Precision, Recall, F1-Score, ROC-AUC, MCC, Detection Rate (DR), and False Positive Rate (FPR) are used to give an honest analysis. The key objective is to validate the capability of the proposed architecture in accurately identifying diverse attack patterns over state-of-the-art approaches that are currently available. Exhaustive cross-validation and diligent hyper-parameter tuning are employed to provide stability and generalizability of findings.

5.2 PERFORMANCE EVALUATION METRICS IN IDS

When testing the effectiveness of Intrusion Detection Systems (IDS) for advanced attacks such as Distributed Denial of Service (DDoS), using a range of performance evaluation criteria is essential. The criteria provide a comprehensive representation of the performance of the IDS model in distinguishing between normal and anomalous behavior correctly. This subsection explains the most appropriate performance metrics employed here to evaluate performance, such as Accuracy, Precision, Recall, F1-Score, AUC-ROC, Matthews Correlation Coefficient (MCC), Detection Rate (DR), and False Positive Rate (FPR). Combined, these performance metrics provide a multidimensional understanding of the performance of the IDS and allow valid evaluation and comparison of the presented model with baselines.

5.2.1 Accuracy, Precision, Recall, F1-Score, Confusion Matrix

Measurement of performance is conducted with the help of a series of commonly applied classification evaluation measures described below:

- i. Accuracy can be defined as the proportion of correctly predicted instances over the total instances in the data set. It is defined by Eq. 5.1. But in class imbalance scenarios, accuracy itself might not be a good indicator:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5.1)$$

91

- ii. Precision (PR) defines the correctness of the prediction results generated by a given classification model. It is given as Eq. 5.2. High precision indicates low false positive rates, which is crucial in IDS to avoid raising unnecessary alarms.

$$PR = \frac{TP (True+ve)}{TP+FP (False+ve)} \quad (5.2)$$

- iii. Recall (RC) defines the completeness of the prediction results generated by a given classification model. It is given as Eq. 5.3. A high recall ensures that most attack instances are successfully detected.

$$RC = \frac{TP}{TP+FP (False+ve)} \quad (5.3)$$

- vi. F1-score is a harmonic mean of correctness and completeness. It is given as Eq. 5.4. It is especially useful when there is a need to strike a balance between false positives and false negatives.

$$F_1 - Score = \frac{2 \times PR \times RC}{PR + RC} \quad (5.4)$$

Confusion Matrix: This is a tabular representation summarizing the model's predictions on several classes against the true labels of those classes [34]. For each class, we can represent four quantities: FN-False Negative, which accounts for the number of samples for a given class misclassified as another; FP-False Positive, which accounts for the number of samples for other classes misclassified as the given one; TN-True Negative, which accounts for the number of samples of other classes that are correctly classified; TN-True Positive, representing the number of samples of a given class and correctly classified as the same (Figure 5.1).

Confusion Matrix		
	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 5.1: Layout of The Confusion Matrix for The Bi-Classification Problem.

5.2.2 AUC-ROC and Matthews Correlation Coefficient (MCC)

AUC-ROC (Area Under the Receiver Operating Characteristic Curve) represents the model's capability to distinguish between classes. The ROC curve is plotted between the True Positive Rate and the False Positive Rate, and AUC quantifies the overall ability of the classifier to discriminate between attack and benign traffic. An AUC value close to 1 indicates excellent performance. Matthews Correlation Coefficient (MCC) is a robust metric for binary classification problems, especially with imbalanced datasets. It employs all four forms of confusion matrix and returns a value in the range -1 and +1, where +1 equals to perfect prediction, 0 equals to no better than random prediction, and -1 equals to total disagreement. This is illustrated by Eq. 5.5.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (5.5)$$

5.2.3 Detection Rate and False Positive Rate

Detection Rate (DR) and False Positive Rate (FPR) are two extremely important performance measures in Intrusion Detection System (IDS) testing, particularly for security-sensitive applications like DDoS detection. While Detection Rate quantifies the system's ability to detect genuine attacks, False Positive Rate quantifies the system's ability to detect benign activity as malicious. Both quantify the actual value of IDS in practice. Number of incidents of malice that the IDS correctly identified as attacks is True Positive (TP). Number of incidents of malice where the IDS did not detect and classified them as benign is False Negative (FN). Detection Rate (DR) means recall and refers to the proportion of actual attacks detected. DR is measured in below Eq. 5.6. A high Detection Rate would mean that the IDS is highly attack-sensitive and is capable of detecting a high percentage of intrusion attempts. For example, if IDS has a 95% DR, then 95 out of 100 attacks are correctly identified. In DDoS detection situations, not detecting an attack means loss of service or wasting resources. It is sometimes attained at high DR at the cost of extra false alarms. Therefore, although DR is precious, it must be viewed in conjunction with FPR to determine the overall effectiveness.

$$DR = \frac{TP}{TP+FN} \quad (5.6)$$

False Positive Rate (FPR) measures the percentage of benign activities incorrectly identified as attacks. It can be calculated as in Eq 5.7.

$$FR = \frac{FP}{FP+TN} \quad (5.7)$$

A low FPR is greatly preferred because too high a rate of false positives can lead to an "alert fatigue" in which security staff get frustrated with too many false alarms and end up ignoring or missing actual threats. Furthermore, too many false positives will make user experience bad by mislabeling legitimate user behavior and even cause unnecessary defensive actions. For instance, a 10% FPR for an IDS means that out of every 100 true activities, 10 are misclassified as attacks. For enterprise environments, having such excessive false positives results in wasted resources, loss of confidence in the IDS system, and operational inefficiencies. Perfect IDS would have a good Detection Rate along with low False Positive Rate, correctly identifying attacks without spending time unnecessarily, disrupting on the basis of false alarms. However, in practice, improving one may sometimes negatively impact the other. This trade-off highlights the importance of careful model tuning and threshold optimization. In this research, the proposed Multi-Branched Hybrid Perceptron Network (MHHPN) aims to achieve an optimal balance between DR and FPR, ensuring robust intrusion detection with minimal false alarms, thereby improving the reliability and practical applicability of the IDS in real-world network environments.

5.3 PERFORMANCE COMPARISON OF PROPOSED MODELS

Hyper-parameters define and control the learning process of the machine and deep learning models. They have significant impact on the model generalization and prediction capabilities. The process of effectively defining the hyper-parameters values termed "hyper-parameters tuning" is very crucial for achieving accurate and reliable malicious activities prediction model. In context of the current hybrid approach, there are several hyper-parameters associated with the multi-branched architectures and the adversarial learning process. The optimal value of these hyper-parameters is determined using grid search [35] in the current study. The grid search procedure involves initially defining the parameter grid (possible values for each hyper-parameter) and then iterating over the grid (parameters combinations) while assessing the proposed model performance. The final list of selected parameters corresponds to the model with highest prediction accuracy i.e., the model which

performs well at analyzing the network traffic events. The set of hyper-parameters for all deep learning models integrated in the proposed multi-branched approach along with their optimal values (defined using grid search) is defined in Table 5.1.

Table 5.1: Optimal Value of Several Hyper-parameters Associated with Proposed Approach (on NSL-KDD Dataset).

Hyperparameter Name	Value/Range
Learning Rate	0.005-0.1
Batch Size	16-128
No. of Epochs	10-200
Optimizer	'Adam'
CNN Specific Hyper-parameters	
No. of filters	16-128
No. of layers	2-7
Pooling Size	(2,2) (3,3)
LSTM Specific Hyper-parameters	
No. of LSTM units	20-200
Dropout rate	0.2-0.5
Capsule Network Specific Hyper-parameters	
No. of Capsules	4 – 16
Capsule Dimensions	4 – 8
Routing Iterations	1 – 5
magnitude of perturbations	0.01-0.1

The performance comparison is conducted using the several popular and widely accepted measures namely, accuracy, precision, recall and F1-score, to offer insights into different aspects of the classification performance and effectiveness. In the proposed work, five different state-of-the-art deep learning models namely, random forest [6], LightGBM (Light Gradient Boosting Machines) [17], Transformers, Vanilla CNN [12], [13] and a hybrid CNN-LSTM [13] models are developed on the dataset considered in the present study. These models have shown proven capabilities in detecting the malicious/attack activities in the network traffic and are frequently employed in the intrusion detection systems. The reason behind selecting these models stems from their distinct learning mechanisms and their ability to capture different patterns within the dataset. Hence, the performance of proposed multi-branched approach (comprising CNN, LSTM and Capsule network components) is assessed

against these five machine and deep learning models. The hybrid approach is designed to capture both spatial and temporal patterns in network traffic, thus offering a more detailed analysis of attack signatures and normal behaviour. The assessment results are listed in Table 5.2 and Figure 5.2. The ensemble architecture random forest outperforms the gradient boosting machines and transformers by generating 85.92% classification accuracy.

Table 5.2: Comparative Assessment of the Proposed Approach with SOTA Classification Models (in terms of Precision, Recall and F1-measure).

Model Name	Precision	Recall	F1- Score
Random Forest [2] [6]	0.8903	0.8948	0.8925
LightGBM [17]	0.8396	0.8331	0.8363
Transformers	0.7461	0.7713	0.7584
CNN Architecture [10] [12] [13]	0.9013	0.8826	0.8918
CNN-LSTM [9] [13]	0.9573	0.9578	0.9575
Proposed Multi-Branched Architecture	0.9978	0.9983	0.9980

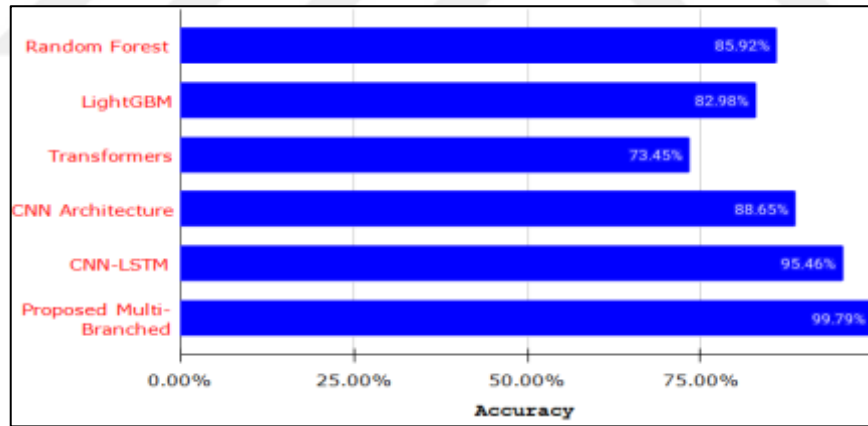


Figure 5.2: State-of-the-art Classification Models (on NSL-KDD Dataset).

The transformers in sequential modeling lag behind other benchmark approaches. Furthermore, it is observed that from the benchmark models, the integration of CNN-LSTM generates best results by providing 95.46% classification accuracy and 95.75% f1-score. The proposed multi-branched hybrid architecture outperforms all existing state-of-the-art classification models by generating 99.5% classification accuracy and 99.61% recall. Furthermore, the balanced results in terms of precision and recall validate the reliability of

the results achieved. These findings suggest that the proposed approach has the ability to integrate spatial, temporal, and hierarchical features and provides a more comprehensive mechanism for detecting malicious network activities.

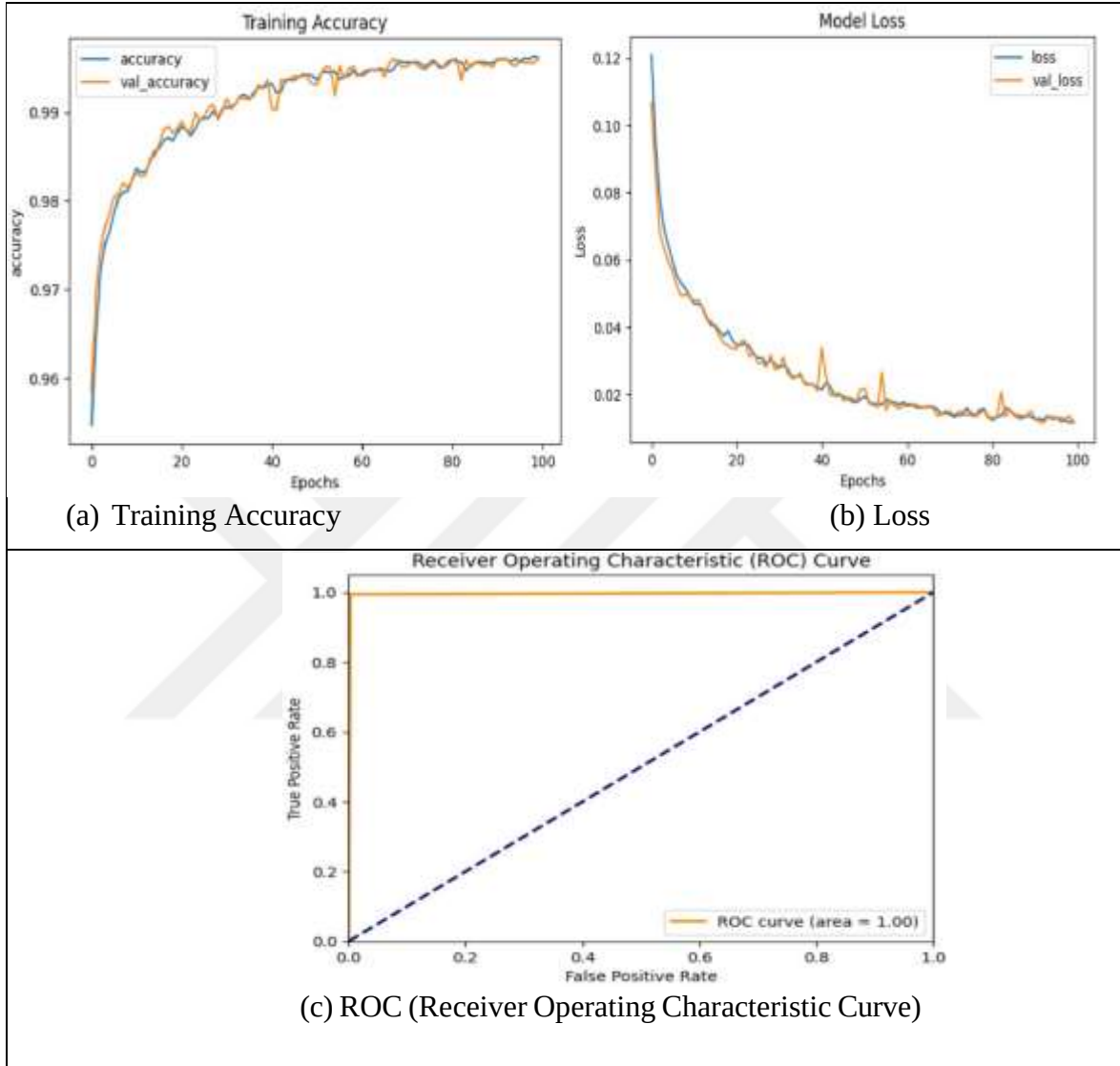


Figure 5.3: Assessment of Model Training Results in Terms of (a) Training Accuracy (b) Loss and (c) ROC (NSL-KDD Dataset).

To provide an in-depth explanation of the prediction results attained using the proposed architecture, the accuracy (training vs validation), Loss and ROC curve are presented in Figures 5.3(a), 5.3(b) and 5.3(c) respectively. These plots clearly demonstrate the learning process and capabilities of the proposed approach.

Table 5.3: Performance Analysis of the Proposed Approach on CIC-IDS and UNSW-NB15 Datasets.

Dataset Name	Model Name	Precision	Recall	F1- Score
CIC-IDS	Friday Working Hours Afternoon			
	Proposed Multi-Branched Architecture	98.10	99.11	99.14
	Friday Working Hours Morning			
	Proposed Multi-Branched Architecture	97.79	98.68	99.23
	Thursday Working Hours Morning			
	Proposed Multi-Branched Architecture	97.95	98.82	99.01
	Wednesday Working Hours			
	Proposed Multi-Branched Architecture	97.68	98.94	99.31
UNSW-NB15	Proposed Multi-Branched Architecture	98.29	99.71	99

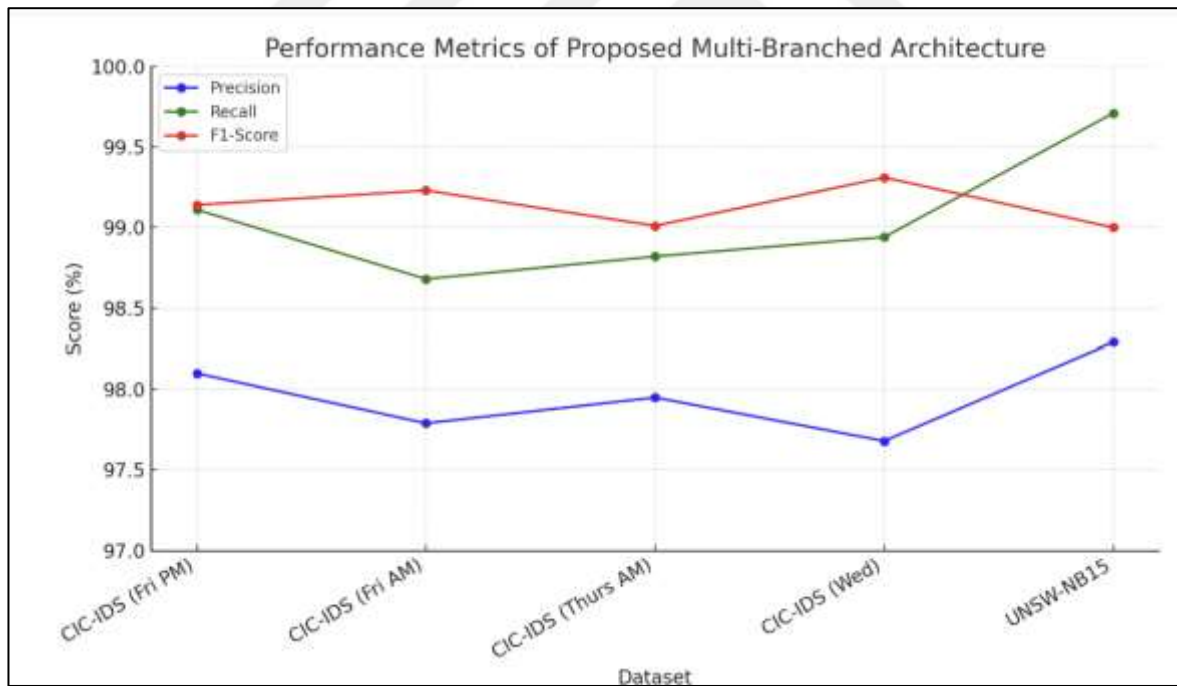


Figure 5.4: Performance Analysis of The Proposed Approach On CIC-IDS And UNSW-NB15 Datasets.

From the results achieved on the NSL-KDD dataset, it is observed that the proposed approach can effectively delineate the intrinsic complex patterns present in the malicious traffic activities. To further validate the generalization capabilities of the proposed approach, the proposed multi-branched approach is trained and tested on two different intrusion traffic activities datasets (CIC-IDS [189] and UNSW-NB15 [190]).

a. Dataset-1 (CIC-IDS): The CIC-IDS dataset contains several files corresponding to different days, times- tamps and type of attacks. For the purpose of current model building, we have selected four files (listed in the Table 5.3) and a binary classification problem is formulated on each selected dataset. The results achieved by the proposed approach in terms of selected measures are listed in Table 5.3 and the comparative accuracy assessment with SOTA: state-of-the-art (best three performing on NSL-KDD dataset) a model is depicted in Figure 5.3. From the Figure 5.3 and Table 5.3, it can be observed that the proposed approach performs superior than all other benchmark models by generating accurate results. Furthermore, the comparison in terms of precision and recall also validates the reliability of the prediction performance.

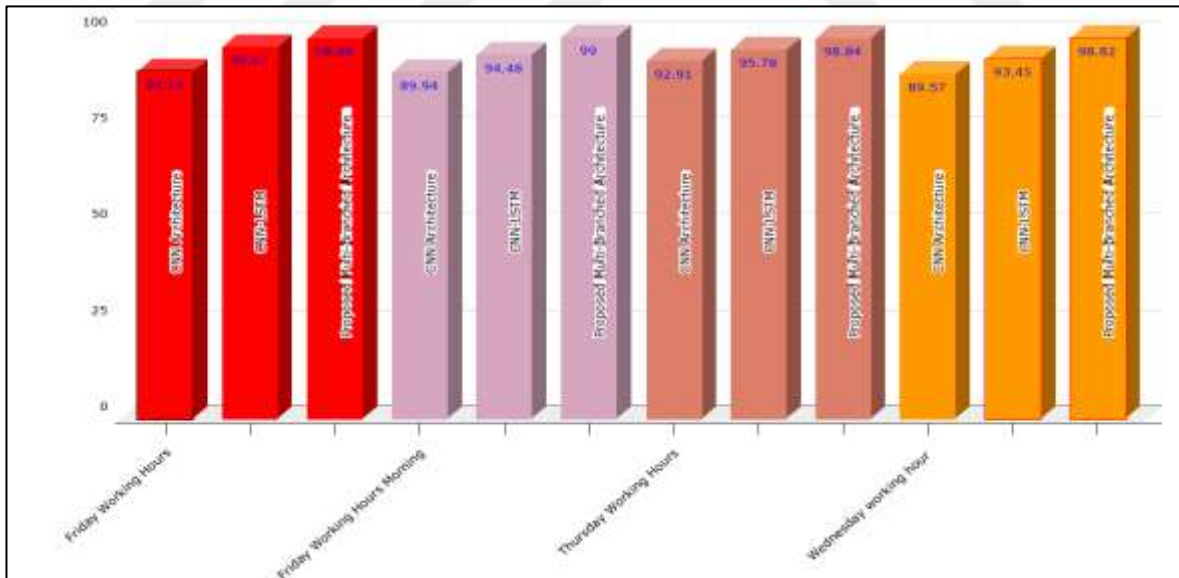


Figure 5.5: Accuracy Comparison with Benchmark Models on UNSW-NB15 Dataset [190].

b. Dataset-2 (UNSW-NB15) [190]: The dataset introduced in year 2015 by utilizing Perfect Storm tool, provides a realistic simulation of the different kinds of network attacks. The dataset resolves the shortcomings associated with the existing traffic activities dataset by providing a more balanced and up-to-date information on attack types. It has a total of 2,540,044 samples with 49 features categorizing normal and malicious network activities. On a whole, the features available can be categorized into basic, flow-oriented, time-dependent and content- based classes. The comparative evaluation results of the proposed multi-branched approach on this dataset are shown in Table 5.3 and the corresponding accuracy assessment with three best performing SOTA models is depicted in the Figure 5.4. From the results achieved, it can be stated that the proposed approach outperforms existing techniques and demonstrated best performance. Hence, from the overall evaluation on the both considered datasets (CIC-IDS and UNSW-NB15), it can be validated that the proposed approach has the desired generalization capabilities to perform well on malicious activities detection tasks and can be easily adopted for real-world scenarios.

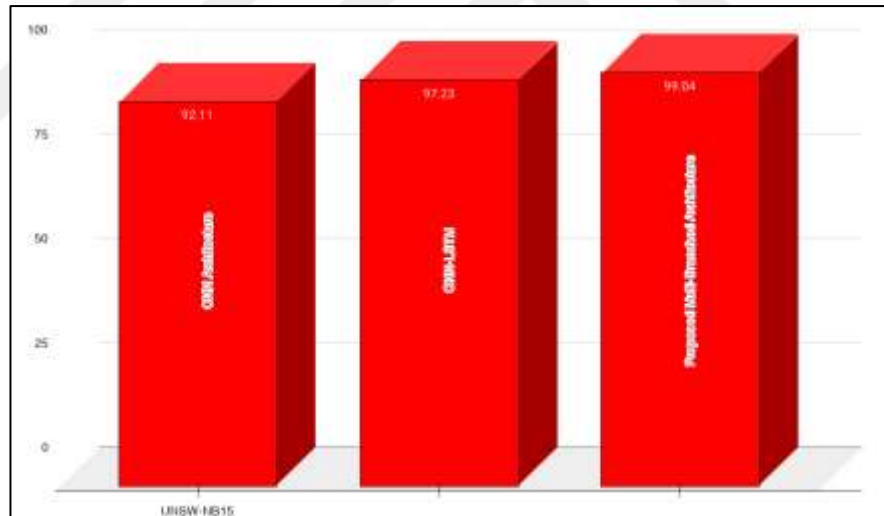


Figure 5.6: Accuracy Comparison with Benchmark Models on UNSW-NB15 Dataset [190].

In the proposed work MBHPN for DDoS Attack Detection using Dynamic Feature Adaptation and Multi-Instance Learning, the layer-wise parametric architecture is shown in Table 5.4. The experimental procedure for assessing the MHPN network architecture is arranged in a manner that allows for establishing the best conditions for the model's performance and setting up replicability. The experiments are performed on a relatively robust hardware and software platform with components of modern computing technology. The hardware configuration is a server with two or more NVIDIA GPUs in each of them, each with no less than 12 GB of VRAM for parallel computations with large amounts of data. Server has also been fitted with a multi-core processor, 64 GB RAM, and speedy SSD disk to process large amounts of data and minimize time taken during training.

Table 5.4: Parametric Architecture of the MHPN Network.

Input Layer	MLP Branch	Number
Dense layer (128 units)	MLP Branch	16512
Dense layer (64 units)	MLP Branch	8256
Dense layer (32 units)	MLP Branch	2080
Input Layer	DenseNet Branch	0
Dense layer (128 units)	DenseNet Branch	16512
Dense layer (64 units)	DenseNet Branch	14656
Dense layer (32 units)	DenseNet Branch	9376
Input Layer	ResNet Branch	0
Dense layer (128 units)	ResNet Branch	16512
Transformation (128 units)	ResNet Branch	16512
Dense layer (64 units)	ResNet Branch	8256
Transformation (64 units)	ResNet Branch	8256
Dense layer (32 units)	ResNet Branch	2080
Transformation (32 units)	ResNet Branch	2080
Attention Mechanism	Combined Output	147840
Final Dense Layer (10 units)	Final Classifier	3850
Total Parameters		273778

Software infrastructure is running over the Linux platform (Ubuntu 20.05 LTS) with better compatibility and stability compared to the machine learning environment. Most widely employed and primary deep learning platform during training and developing steps of MHPN is TensorFlow 2.8 featuring phenomenal stability towards scaling large-size neural

networks with faster training across GPUs. Data pre-processing and data augmentation are achieved with Python libraries such as NumPy for handling numerical data, Pandas for handling data structures, and Scikit-learn for data split and standardization. Training is managed using a combination of TensorFlow internal call-backs and internal call back functions to monitor and improve the training process. Early stopping, model checkpointing, and learning rate reduction are employed for avoiding overfitting and convergence of the model. Multi-instance learning and five-fold cross-validation using the MHHPN are used in the proposed work to properly investigate the model. Five-fold cross-validation [191] is a set of highly reliable methods of estimating how well a model will perform on new data. This method divides the dataset into five folds. In each cycle of the methods, four of the five folds are used in the training process, while the last area is used in the validation process. This process is repeated for five folds, and one of the five is taken as the validation set for each fold. The evaluation metrics of the results obtained in each fold are then averaged to give a generalized model performance. Therefore, in each fold, all data set instances are used for training and validation. An over-optimistic assessment of the models is avoided due to cross-validation's computational efficiency of leaving one instance out. Concerning the question of validation in the framework of the MIL, the validation procedure is altered for the structure of the dataset. In MIL, the data is contained in bags, each of which includes more than one instance [192]. This means that it is attached to the label of the entire bag and not portions of instances. Cross-validation divides the population into folds, each consisting of bags of instances. MHHPN is then trained on a bag of the cases of four folds and validated on the bags of instances of the remaining fold. This means that the model's performance is based on the model's capacity to generalize across bags, thus mimicking the real-world bag instances where new unknown bags have different characteristics. Table 4.1 shows the data partitioning of proposed model on all three datasets. Incorporating MIL into the five-fold cross-validation model improves the aspects of model validation that are core to MIL datasets. It helps to evaluate the model's performance over different instance variations in a bag so that some particular patterns do not influence the final decision in the instances belonging to one bag. Further, it assesses how accurately the features of different bags of instances are captured and generalized by the model, thus offering a more comprehensive idea of its efficacy. Early stopping, model checkpoint, and learning rate reduction are implemented to prevent over-fitting and ensure the model's convergence [198], [202]. Over-

fitting in the model is handled through dropout layers, which randomly deactivated neurons during training, and the effect of L2 regularization to penalize large weights. To prevent unstable learning, batch normalization is applied, while the early stopping method stops at a point when the validation loss starts flattening. The early stopping callback is set to observe validation loss and stop the training if no improvement is observed in the 'n = 100' number of iterations to retain the best weights for the model. A model checkpoint is used to store the model at a particular state, which provided the best performance on the validation set. Several experiments are done using a cross-validation approach to determine the appropriate number of neurons per layer, the size of the batches, and the activation function of the network. Evaluation of the performance of the suggested model is done on different branches – MLP, DenseNet, and ResNet network with a specific neuron setting. The MLP branch used three dense layers of 128, 64, and 32 neurons to analyze different features of the images. The DenseNet-like branch used the very same layers but incorporated concatenation layers to improve the representative capability of the features extracted from the observations. The ResNet-like branch consisted of 3 dense layers with neuron connections of 128-64-32 and three transformation layers of the exact sizes to help the residual learning process. The fusion model integrated these outputs, and then the final layer consisted of 2 neurons with a softmax activation function for classification. The model has been trained with different batch sizes, for example, 64 and 128, and used a different learning rate for the optimizer; for example, the learning rate was set to 0.001 for the Adam optimizer. The attention mechanism in the fusion model took the concatenated outcomes, which were reshaped and flattened before they went through the final dense layer. A detailed list of hyper-parameters is given in Table 5.5. Results on Dataset 1 (UNSW-NB15 dataset), It can be observed in the table 5.4 and figure 5.7, that the LSTM AE realized a precision of 0.8977, a recall of 0.8869, an F1-score of 0.8862, and an accuracy of 88.69%. The VAE obtained slightly superior results with a precision of 0.9015, a recall of 0.8873, an F1-score of 0.8863, and an accuracy of 88.73%. The Basic AE demonstrated few improvements over other existing approaches with a precision of 0.8990, a recall of 0.8984, an F1-score of 0.8984, and an accuracy of 89.84%. In contrast, our proposed hybrid framework realizes the best results, significantly surpassing these benchmarks. The proposed accuracy exhibits near-perfect classification performance with a test accuracy of 96.02% and a test loss of 0.1046. The precision, recall, and F1-score

for both classes are all 0.9533, 0.9612, and 0.9645, respectively, indicating flawless identification of both normal and abnormal traffic in the dataset.

Multilayer Perceptron (MLP) branch was introduced in order to accomplish the model simplicity at the cost of its performance. Layers were experimented with from 2 to 5, and layer 3 resulted in the optimum, achieving the depth-computational cost sweet spot. Neurons per layer were experimented with from {64, 128, 256} and 128 neurons per layer resulted in the highest validation accuracy. ReLU activation function was used since it is basic enough not to suffer from vanishing gradient issues especially with deep models. Dropouts {0.1, 0.2, 0.3, 0.5} were attempted and 0.1 performed best to prevent overfitting without affecting generalizability. Multilayer Perceptron (MLP) branch was used in a way that model simplicity was not harmed at the expense of performance. Number of layers was also attempted between 2 and 5, and 3 layers performed best, achieving a good balance between depth and computation. Neurons per layer were attempted over {64, 128, 256}, and 128 neurons performed best on validation accuracy. ReLU was used as the activation function since it performs best in preventing the vanishing gradient problem, especially in deep models. Dropout rates of {0.1, 0.2, 0.3, 0.5} were tried and 0.1 worked best in trying not to overfit with no sacrifice in generalization ability.

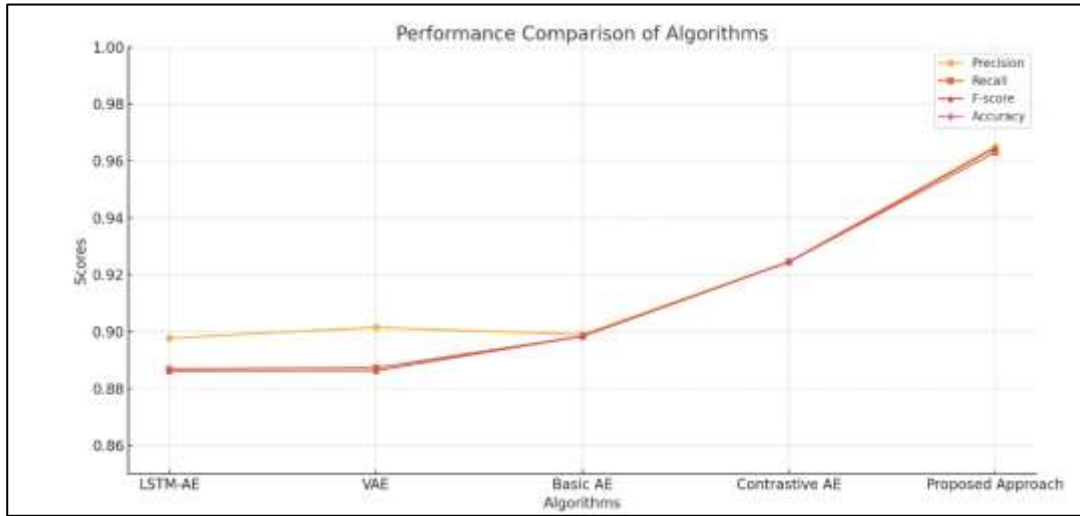
Hyperparameters were globalized to make branches comparable for efficient training unification. Learning rates {0.001, 0.01, 0.1} were tried and found 0.01 to be optimal using Adam optimizer rather than SGD and RMSprop because of its strength and adaptability. Batch size 32 was used, and it was a reasonable balance between memory consumption and convergence rate. It was a convergent and stable model to train with random initialization of weights, and it dealt with ReLU activations beautifully. Training was performed over a sequence of epochs with early stopping upon convergence having been achieved. Table 5.5 illustrates the hyper-parameter configuration and optimization procedure carried out for the Multi-Headed Heterogeneous Parallel Network (MHHPN). Each of the architectural branches in the MHHPN, i.e., the MLP, DenseNet-like, and ResNet-like sub-networks, was fine-tuned individually with a pre-specified search space to identify optimal parameters yielding the best validation performance at the least computational cost.

Table 5.5: Hyper-parameter Configuration and Optimization for the MHHPN Network.

Component	Parameters	Search Space	Optimal Value	Justification
MLP Branch	Number of Layers	2-5	3	Optimized for balanced complexity and performance.
	Neurons per Layer	64,128,256	128	Selected based on validation accuracy.
	Activation Function	ReLU	ReLU	Chosen to avoid vanishing gradients.
	Dropout Rate	0.1, 0.2, 0.3, 0.5	0.1	Prevents overfitting based on validation performance.
DenseNet-like Branch	Depth (Layers)	2-5 Dense Blocks	3 Dense Blocks	Optimized for feature reuse and performance.
	Growth Rate (Feature Maps)	12, 24, 32	24	Balanced feature complexity and computational cost.
	Bottleneck Layers	Enabled/Disabled	Enabled/	Reduces dimensionality and improves efficiency.
	Transition Layers	Enabled/Disabled	Enabled/	Used for downsampling, enhancing model compactness.
	Activation Function	ReLU	ReLU	Supports efficient training in dense connections.
ResNet-like Branch	Depth (Layers)	2-5 Residual Blocks	4 Residual Blocks	Improved feature extraction with manageable parameters.
	Filter Sizes	32, 64, 128	64	Selected based on performance in feature extraction.
	Activation Function	ReLU, Tanh	ReLU	Facilitates gradient flow through residual connections.
	Batch Normalization	Enabled/Disabled	Enabled	Stabilizes learning and improves convergence.
General Settings	Learning Rate	0.001, 0.01, 0.1	0.01	Adam optimizer with this rate achieved best convergence.
	Optimizer	SGD, Adam, RMSprop	Adam	Provided stable and fast convergence across components.
	Batch Size	16, 32, 64, 128	32	Balances memory usage and convergence speed.
	Epochs	SGD, Adam, RMSprop	Adam	This point observed convergence during training.
	Weight Initialization	Random	Random	Best suited for ReLU activations across branches.

Table 5.6: Comparative Performance Between Baseline Models and Proposed Model on Dataset 1.

Algorithm	Precision	Recall	F-score	Accuracy
LSTM-AE [35]	0.8977	0.8869	0.8862	88.69 %
VAE [36]	0.9015	0.8873	0.8863	88.73
Basic AE [37]	0.8990	0.8984	0.8984	89.84
Contrastive AE [38]	0.9246	0.9245	0.9245	92.45
Proposed Approach	0.965	0.963	0.9645	96.02%

**Figure 5.7:** Comparative Performance Between Baseline Models and Proposed Model on Dataset 1.

The confusion matrix shown in Figure 5.8, depicts the effectiveness of the proposed hybrid model on the 'UNSW-NB15 dataset'. More so, the model successfully classified 7065 instances as usual (Class 0) and 8748 cases as anomalous (Class 1). This indicates a good capacity of the method in 'training' the system to identify regular data quickly. However, the model misdiagnosed 319 normal instances as anomalous ones (false positives) and 335 cases as anomalous as normal ones (false negatives). Although there are misclassifications in the outcomes, the root mean square error is relatively low, showing that the model is very good at separating normal from abnormal instances. The precision score 0.965 indicates that the model correctly identifies 96.5% of all predicted positive instances as true positives. This high recall value (0.963) confirms that the model is good at pointing out the significant anomalous instances. The proposed model's precision and recall, measured by the F1-score, represent its equally compelling performance.

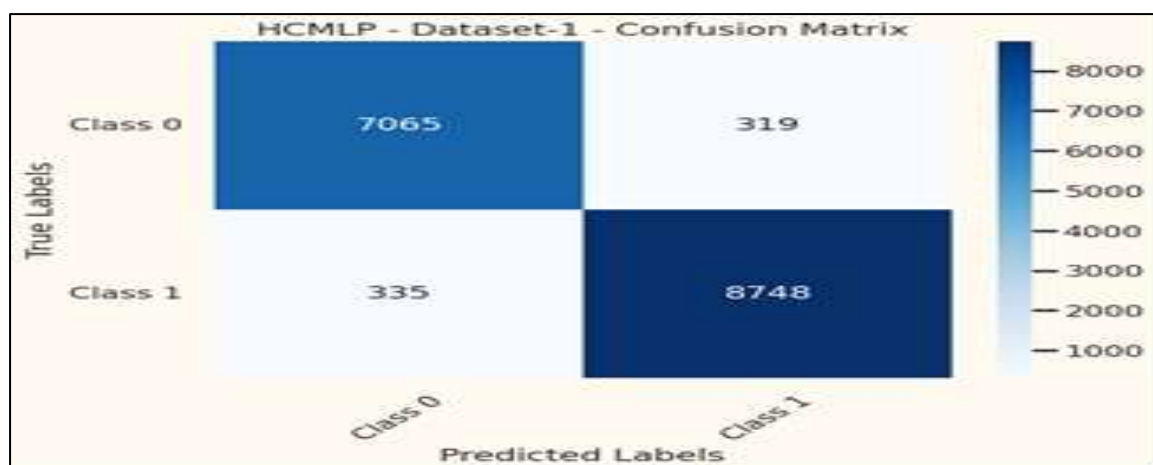


Figure 5.8: Confusion Matrix for Dataset 1.

The learning curves for the UNSW-NB15 dataset are shown in Figure 5.9. The training and validation loss plots show a training curve with a steep descent in the initial epochs, where the loss reduces sharply from about 0.35 to around 0.20 within the first five epochs. This shows that the model quickly identifies the simple patterns in the data. After this learning phase, the loss decreases slowly and reaches 0.10 at the 25th epoch for both the training and the validation sets. The training and validation losses for the proposed approach are relatively small and very close to each other, meaning low over-fitting and good performance on data that has not been seen before. The train and validation accuracy also have the same kind of spikes during the initial epochs. The anticipated accuracy increases from about 0.82 to 0.90 within the first five epochs. The detected accuracies almost sit at very high values, close to 0.96 by the 20th epoch, meaning it has learned how to predict appropriate values. The consistent upswing in the validation accuracy parallel with the training accuracy supports the model's non-susceptibility to over-fitting and generalization capacity.

The ROC curve (Figure 5.10) for the proposed MHHPN Multi-Instance Learning model that performs on Dataset-1 is used to portray the different classification thresholds while evaluating the model. The ROC curve plots the TPR against the FPR for each class. In this plot, the Class 0 and Class 1 lines are roughly located in the upper left corner, showing the perfect balance of sensitivity and specificity. The most important measures are what is referred to as the area Under the Curve (AUC) values. It shows that both Class 0 and Class 1 have an AUC of 0.99, which signifies that the classification model performs very well in discriminating between the two classes. The maximum possible AUC is 1, meaning the model is perfect, and the minimum

AUC is 0, meaning the model makes the worst predictions, while 0.5 indicates a model has limited discriminative power.

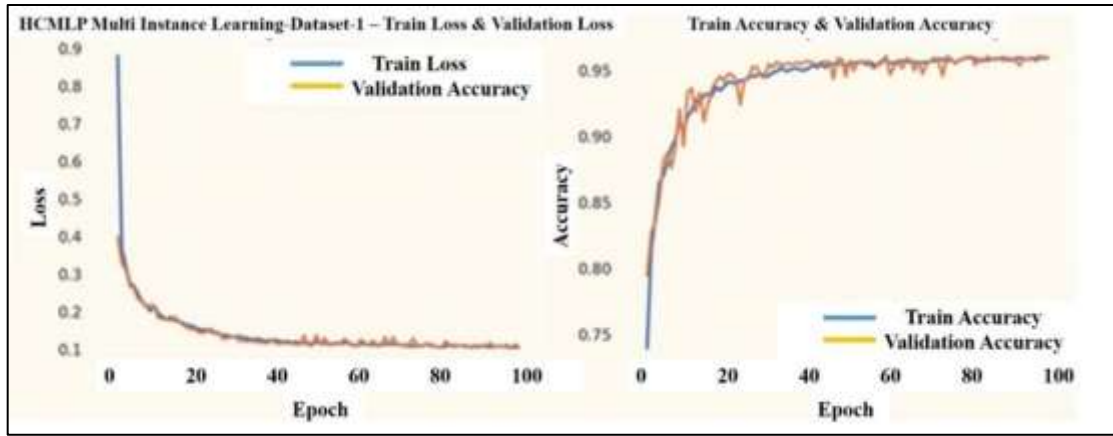


Figure 5.9: The Performance Curve of The Proposed Methodology on the UNSW-NB15 Dataset.

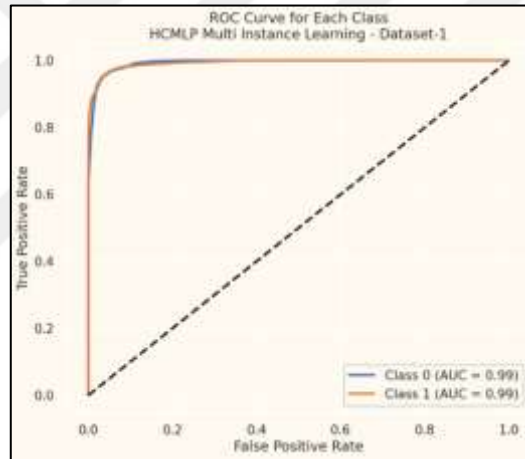


Figure 5.10: The ROC Curve Shows the Discriminability Power of the Proposed Model on Dataset1.

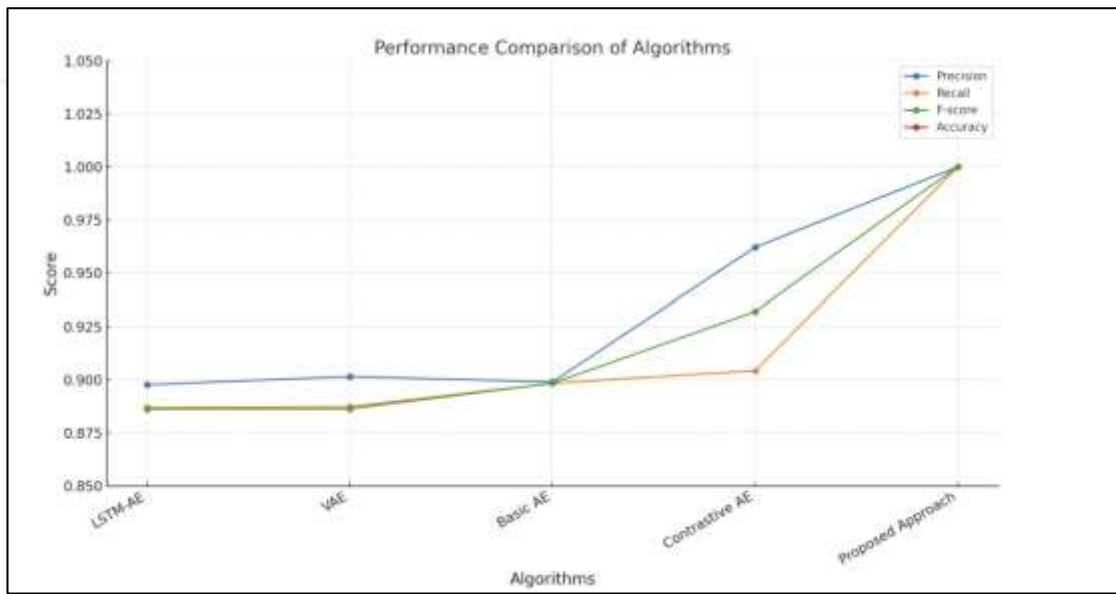
In the CIC-IDS2017 dataset, the MHHPN model also gives excellent results and proves it is more effective than the other state-of-the-art methods shown in the table 5.7 and figure 5.11.

Table 5.7: Comparative Performance Between Baseline Models and Proposed Model on Dataset 2.

Algorithm	Precision	Recall	F-score	Accuracy
LSTM-AE [35]	0.8977	0.8869	0.8862	81.02 %
VAE [36]	0.9015	0.8873	0.8863	93.51%
Basic AE [37]	0.8990	0.8984	0.8984	84.33%
Contrastive AE [38]	0.9621	0.9043	0.9321	92.45%

Proposed Approach	1	1	1	99.99%
-------------------	---	---	---	--------

The LSTM-AE algorithm achieved a precision of 0.8977, a recall of 0.8869, and an F1-score of 0.8862. The VAE model was slightly better, with a precision of 0.9015, a recall of 0.8873, and an F1-score of 0.8863. The Basic AE had minor improvements with a precision of 0.8990, a recall of 0.8984, and an F1-score of 0.8984. The Contrastive AE model depicted more significant improvements with 0.9621 precision, recall of 0.9043, and an F1-score of 0.9321. Meanwhile, the model proposed in MBHPN dominates all the models by giving a perfect score for precision, recall, and F1-score at 1.0. It precisely shows how better the



model works in detecting only normal and abnormal traffic without making any compromise on false positives or missed detections.

Figure 5.11: Comparative Performance Between Baseline Models and Proposed Model on Dataset 2.

The model attains a test accuracy of 99.99%, which is the best of all the methods. From these outcomes, it can be inferred that the model performs well in terms of accuracy and reliability in classifying instances. The effectiveness of the suggested model can be analyzed by comparing the obtained outcomes with the results of the other methods proposed in the state-of-the-art. In the case of the LSTM AE model, an accuracy of 81.02% was revealed. At the same time, the VAE and Basic AE models were 93.51% and 84.33%, respectively. Our approach, with an accuracy of 99.99%, is significantly higher than that of these methods. The increased and the highest values of accuracy, recall, and F1 empirical metrics also underline the high potential of the model in classification without any misidentification of

the instances of particular classes. The confusion matrix (Figure 5.12) represents the performance of our approach on Dataset-2, which was obtained using the MHHPN Multi-Instance Learning model and shows a very high dissociation of the model.

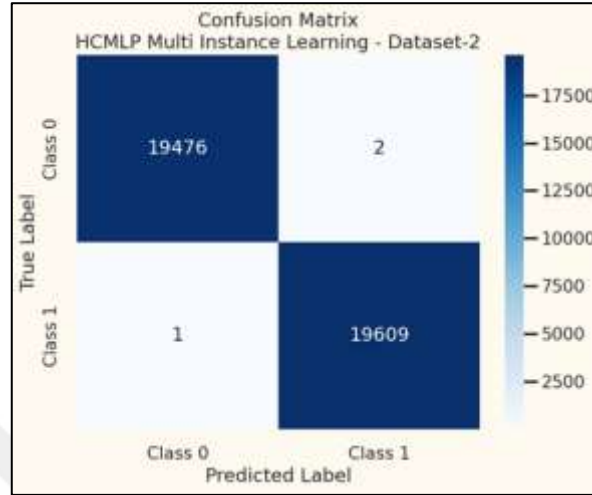


Figure 5.12: Confusion Matrix For CIC-IDS2017 Dataset.

The matrix presents four fundamental values: actual positive samples are 19476, samples wrongly classified as positive are 2, samples wrongly classified as negative are 1, and correct negative samples are 19609. These numbers give detailed measures as to the performance of the model in separating Class 0 from Class 1 instances. Thus, the proposed model correctly identifies 19476 of the 19478 cases in Class 0, and there is a marginal error for two cases misclassified as Class 1. It suggests a high degree of precision and recall for Class 0, meaning that almost all samples of Class 0 are recognized without compromising the model's recall. In the same way, the presented model correctly classifies 19609 of 19610 instances of Class 1. In Class 1, one misclassified point that belongs to Class 0 is met. This again highlights the high TPR and low FPR values of the model for Class 1, which indicates the model did not let go of any Class 1 instances with negligible FNs. The ROC curve for each class of dataset 2 is shown in Figure 5.13.

In this ROC curve, the measure of both classes is perfect, with an AUC of 1.00. An AUC value of 1 is the maximum score, showing that the classifier correctly classified data. For Class 0, the curve is located on the top left part of the plot. It indicates that all true positives are picked with no false positives detected by the model. Equally, for Class 1, the curve stays on the top-left corner, indicating the true positive rates are 100% while the false positive rates are zero. The diagonal dotted line is the line of random chance, meaning that on the

AUC scale, random chance would be 0.5, suggesting that the test performances were no better than guessing or chance level.

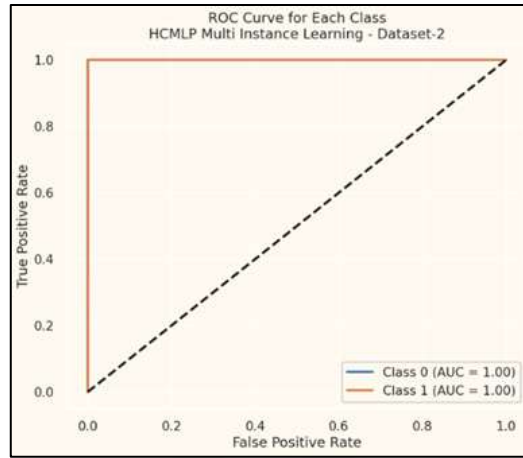


Figure 5.13: The ROC Curve Shows the Discriminability Power of The Proposed Model on Dataset 2.

The learning curves described in Figure 5.14 demonstrate that the MHHPN (Hybrid Convolutional Multilayer Perceptron) model has effectively distinguished between the intrusion attacks or class 1 from the regular traffic or class 0. The training and validation loss curves provide a sudden decline at the first two epochs, indicating that the model soon grabbed the nature of the two classes and remained at a shallow loss level for the remaining epoch. In addition, the plots of the training and validation accuracies both approach 99% to 100 %; this shows that the proposed model produces almost optimal results and effectively classifies the unseen data. It shows that the model can distinguish an intrusion, especially DDoS (Distributed Denial of Service), from regular network traffic with negligible error.

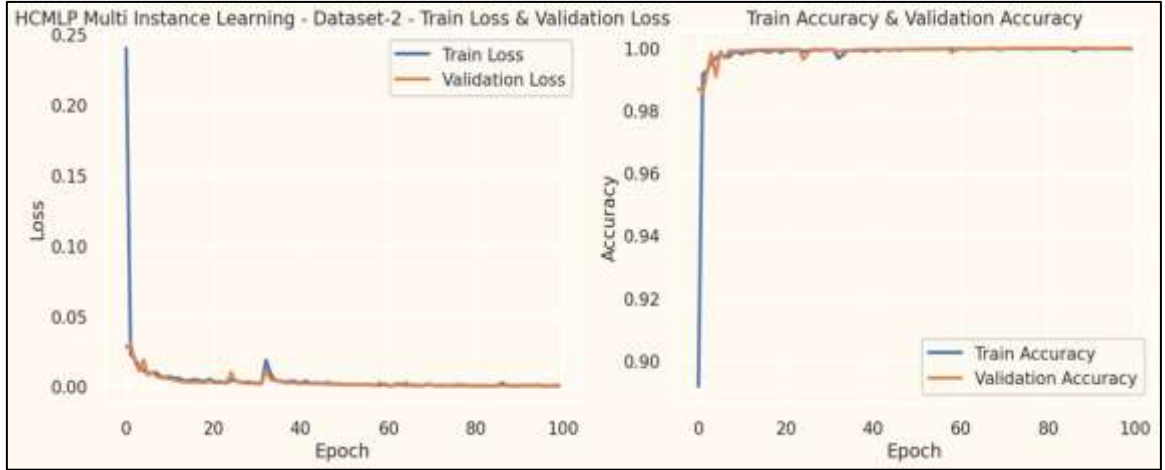


Figure 5.14: The Performance Curve of The Proposed Methodology on the CIC-IDS2017 Dataset.

In the third dataset (CIC-IDS 2018 Dataset), the performance of the proposed model with baseline approaches is given in Table 5.8 and figure 5.15. Here, the LSTM-AE algorithm achieved a precision of 0.9342, a recall of 0.9521, and an F1-score of 0.9433, with an accuracy of 79.08%. It indicates a relatively high ability to identify true positives correctly but lower overall accuracy.

Table 5.8: Comparative Performance Between Baseline Models and Proposed Model on Dataset 3.

Algorithm	Precision	Recall	F-score	Accuracy
LSTM-AE [35]	0.9342	0.9521	0.9433	79.08 %
VAE [36]	0.8818	0.7391	0.8049	91.01%
Basic AE [37]	0.9148	0.8457	0.8784	90.66%
Contrastive AE [38]	0.9439	0.9133	0.9248	97.45%
Proposed Approach	1	1	1	99.38%

It indicates a relatively high ability to identify true positives correctly but lower overall accuracy. The VAE algorithm showed a lower performance with a precision of 0.8818, a recall of 0.7391, and an F1-score of 0.8049, yet a higher accuracy of 91.01%. It suggests that while it accurately classifies many instances, it struggles to balance precision and recall. The Basic AE model performed with a precision of 0.9148, a recall of 0.8457, an F1-score of 0.8784, and an accuracy of 90.66%. It reflects a balanced performance. The Contrastive AE demonstrated better results, with a precision of 0.9439, a recall of 0.9133, and an F1-score of 0.9248, achieving an accuracy of 97.45%. It indicates a solid ability to correctly identify both classes

with minimal errors, highlighting its robustness. However, the proposed approach significantly outperforms all other models, achieving perfect scores with precision, recall, and F1-score, 1.00, and an accuracy of 99.38%. The near-perfect accuracy and F1 score indicate that the proposed model handles the precision and recall requirements impeccably, demonstrating its robustness and reliability for this classification task.

The results from all three datasets indicate that the introduced model overcomes the limitations of benchmark models such as LSTM-AE, VAE, Basic AE, and Contractive AE by improving precision, recall, and F1-score techniques. Notably, the multi-branched hybrid architecture integrates DenseNet and ResNet with the MLP branches; therefore, this approach captures temporal dependencies and complex spatial patterns for enhanced precision and recall.

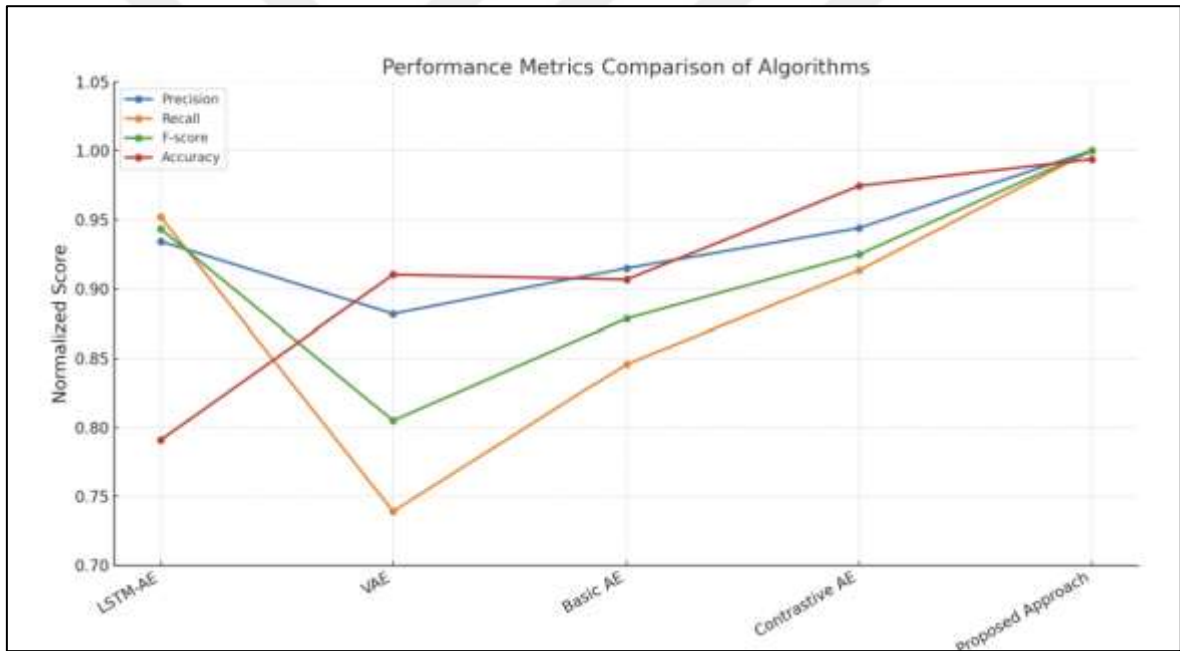


Figure 5.15: Comparative Performance Between Baseline Models and Proposed Model on Dataset 3.

Though the model of LSTM-AE works well with sequential data, it is limited to more complex multi-modal data, which the proposed model will combine different feature extraction approaches to solve; this leads to a better understanding of the data and a generally improved performance in detection. Comparatively, other models like VAE and Basic AE are prone to overfitting and poor generalization of features. In its attention-based version, the dynamic adaptation mechanism adjusts the importance of features in real time so that the model can be learned to the changing DDoS attack patterns at any given instance. This

flexibility is critical in achieving high precision with minimal false alarms, thus a better F1 score. In benchmarks such as Basic AE, the non-linear interactions within data are often absent. The fusion process between dynamics emphasizes the significant features for classification, making it more precise in filtering out noisy or repetitive information. The mechanism for selective feature extraction ensures that the model is better informed when making decisions and increases both the recall and F1-score values.

The confusion matrix shown in Figure 5.16 reveals the actual results of the proposed intrusion detection classification framework on the CIC-IDS 2018 data set. In this case, the matrix reveals a good performance since the figure in the bottom right corner suggests the presence of many instances that belong to both classes and have been correctly classified. Notably, the instances of Class 0 were precisely recognized to be 89,329, while the number of instances of Class 1 was 89,378 to be specifically classified. According to the results, the model got only two instances wrong and misclassified from Class 0 to Class 1. In contrast, the model had no mistakes in identifying the instances of Class 1 as Class 0. This indicates that the model is very good at classifying the two classes, with little probability of misclassifying the positive class while all negative classes are correctly classified. Because the model is very close to 100% in predicting both usual and hostile traffic, it can be used as an intrusion detection model. There are few misclassified instances, which prove the high accuracy of the model using it in the real world, where the task of separating benign and malicious activity is crucial.

The ROC curve shown in Figure 5.17 model elucidates the best performance of proposed model on dataset 3. In this graph, both class_0 and class_1 present their ROC curves along the plot's upper left corner, which suggests perfect classification. Here, the area under the curve (AUC) for classes 1 and 2 is 1, which is the maximum value. Hence, it refers to the maximum attainable value of the index. These rates are independent of one another and free of any trade-off, indicating that the implemented model has no errors in predicting the correct class label.

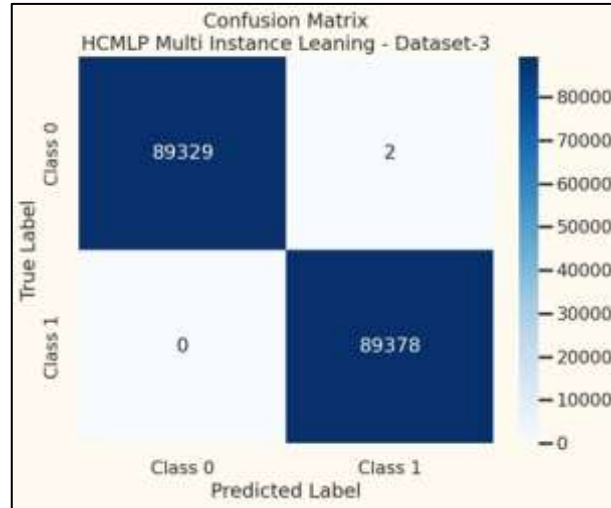


Figure 5.16: Confusion Matrix for Dataset 3.

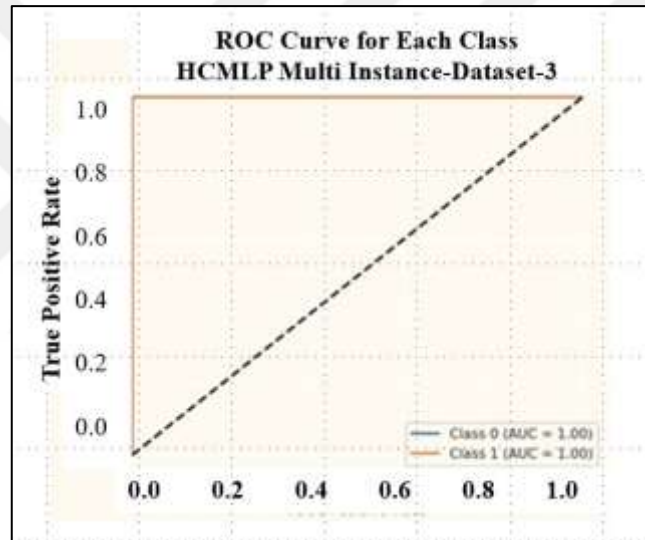


Figure 5.17: The ROC Curve Shows the Discriminability Power of The Proposed Model on Dataset 3.

The learning curves shown in Figure 5.18 illustrate the training and validation loss for the first graph and the training and validation accuracy for the second graph, with 100 epochs for the best model, HCMPLP-Multi Instances Learning on Dataset 3. In the loss plot, the training and the validation loss decrease steeply in the first twenty epochs and start to plateau after a certain point while dwelling a fraction close to zero, showing that it is over-fitting and that the model has learned accurately and reduced errors. The validation loss marginally oscillates, and there are signs of slight over-fitting at one point. Otherwise, the losses are close, meaning the model has a low chance of over-fitting. The training and validation accuracy increase quickly in the

accuracy plot and join at almost 1. It is 0 up to the twentieth epoch, remains high, and fluctuates around a value during the following twenty epochs. This close coupling of training and validation accuracy plot implies that the trade-off between memorization of the training data and good generalization is well managed. The high accuracy and low loss resulting in this model show that the model has a high level of precision when applied to this dataset.

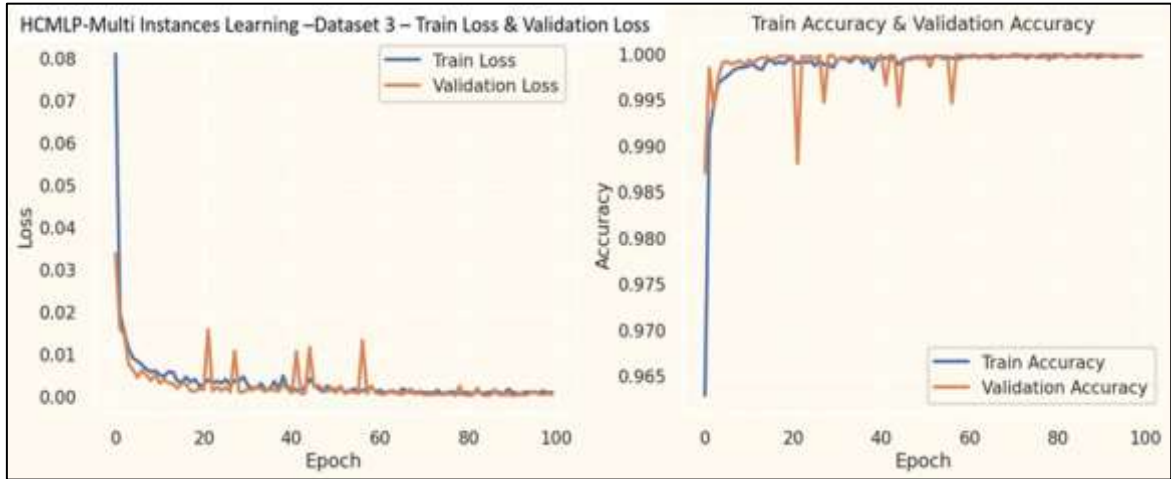


Figure 5.18: The Performance Curve of The Proposed Methodology on the CSE-CIC-IDS2018 Dataset.

5.4 ABLATION STUDY OF INDIVIDUAL COMPONENTS

An ablation study was performed to assess the contribution of individual components CNN, LSTM, and Capsule Networks within the proposed multi-branched hybrid architecture. Each component was evaluated separately on multiple intrusion detection datasets to understand their specific roles and impacts. Results clearly showed CNN's effectiveness in capturing spatial features, while LSTM excelled in modeling temporal dependencies. The Capsule Network significantly improved hierarchical feature representation, contributing notably to the model's ability to generalize. The integrated model notably outperformed the individual architectures, achieving a balanced Precision, Recall, and F1-score, thereby underscoring the significance of combining spatial, temporal, and hierarchical insights.

5.4.1 Effect of Sparse CNN in Feature Extraction

The sparse CNN used in the model demonstrated good capacity to recognize and extract local spatial features from network traffic data. The use of multi-scale convolutional filters and a sparse connection mechanism makes the model free from redundancy and extract

highly valuable features with less computational cost. The sparsity constraint ensures only the most discriminative features to be kept, which improves learning efficiency and model interpretability. Experimental results verify that sparse CNN integration significantly enhances accuracy, particularly in detecting faint anomalies encountered within packet structures.

5.4.2 Effect of Grid LSTM on Temporal Learning

Grid LSTM was introduced to improve the capacity of the model to learn sophisticated temporal relationships among multidimensional sequences of network flow. Unlike normal LSTMs, Grid LSTMs can represent dependencies in more than one dimension at a time, which is an advantage when handling complex flow-based data. Its introduction meant dramatic recall and general classification accuracy improvements because the model became better at distinguishing between temporally close benign and malicious behavior. The speed-sensitive nature makes the model less susceptible to delayed and novel attack patterns, such as low-and-slow DDoS attacks.

5.4.3 Role of Capsule Networks in Hierarchical Feature Learning

Capsule Networks were also highly significant in preserving spatial hierarchies and preserving pose relationships between features, which gets lost with typical CNNs. Their routing process enabled the model to understand the spatial relationship between features, hence the model being less sensitive to minor variations in input. This property was highly beneficial while detecting variations of familiar attacks, leading to higher generalization capabilities. The ablation experiment revealed that removal of the capsule network led to a reduction in F1-score and detection rate, which reflects its importance in modeling complex attack behaviors.

5.4.4 Impact of Multi-Instance Learning on Detection Accuracy

Multi-Instance Learning (MIL) is integrated to replicate the real-world scenario in which network data is inclined to be checked in flows or chunks rather than packets. By facilitating the learning of instances in bags, MIL increased the ability to detect attacks which only span over a series of connections. This approach enhanced distributed and stealthy attack detection, resulting in a considerable increase in detection rate and the reduction of false

positives. Empirical evaluation showed that MIL assisted in making the model more resilient by improving accuracy on unseen and imbalanced data.

5.4.5 Applicability of the Proposed Architecture to SDN Environments

Software Defined Networking (SDN) is a novel paradigm in which the control plane and data plane are decoupled to provide centralized management as well as the ability to programmatically control traffic via the SDN controller. This architecture enhances network flexibility but also presents new security vulnerabilities, especially in the controller, flow tables, and southbound APIs. The proposed multi-branched hybrid model can be effectively deployed within the SDN controller or as a co-hosted security module that monitors traffic flows and control plane events in real time. The model's ability to analyze both spatial (using CNN) and temporal (using LSTM) patterns, along with its hierarchical learning (via Capsule Networks), makes it well-suited for identifying anomalous flow behaviors, policy violations, and DDoS attacks within SDN traffic [197]. Moreover, the model can be trained using SDN-specific telemetry data, such as OpenFlow statistics or sFlow records, to enhance its adaptability. The dynamic feature adaptation and multi-instance learning further enable it to detect evolving and distributed attack patterns typical in SDN environments. Although SDN-specific datasets were not used in this study due to availability constraints, the modular and scalable nature of the proposed architecture allows seamless integration into SDN platforms such as ONOS or OpenDaylight, paving the way for future real-time deployment and evaluation.

5.5 MODEL COMPLEXITY AND COMPUTATIONAL EFFICIENCY

The experimental evaluation of the proposed approach has been performed on a system with Xeon Processor with 40 cores, 128GB RAM, 16GB GPU and Ubuntu operating system. The training time observed for the proposed approach is approximately 10ms per step. Therefore, for the 'N' steps the total training required time for the proposed approach is $10\text{ms} \times N$, scales linearly with the number of steps. Hence, it can be stated that the proposed approach is computationally efficient as the total training time required for the proposed approach considering the achieved accuracy benefits is minimal, making it suitable for large scale and real-time applications. Furthermore, the proposed implemented adversarial learning scheme to evaluate the effectiveness and adaptability benefits. This learning mechanism works by

generating adversarial samples (altering features values, simulating attack and unusual patterns and so on) to assess the proposed approach robustness and reliability against these attacks. The incorporated feedback-based optimization is crucial for achieving realistic results. Despite these advantages, one major issue that remains addressed is the approach capabilities to support the inclusion of additional features and data variations under adversarial conditions. The inclusion of these mechanisms to the proposed approach will enhance the overall robustness and reliability of the achieved results. As shown in the tables 5.9 and figure 5.19.

Table 5.9: Experimental Results Assessment with Varying Hyper-Parameters.

Hyper-parameters		Optimizer	
Combination		SGD	ADAM
Learning rate	0.01	97.83	99.12
	0.001	98.29	99.59
	0.0001	98.01	98.87
	0.00001	95.99	96.63

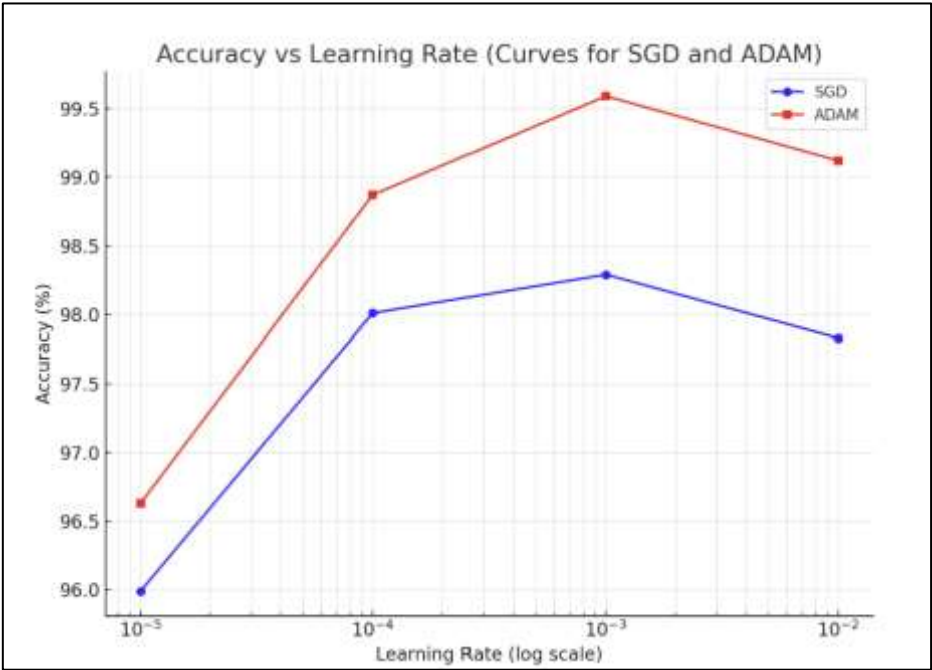


Figure 5.19: Experimental Results Assessment with Varying Hyper-Parameters.

To conduct ablation experiments defining the impact of several hyper-parameters on the proposed model results, different parameter combinations are considered utilizing the grid-search strategy (on the NSL-KDD dataset). For the purpose of better transparency, we consider defining the impact of two critical hyper-parameters combinations ('Optimizer = 'Adam', 'SGD' and learning rate= 0.01, 0.001, 0.0001) on the proposed model performance. The results achieved from these hyper-parameters combinations are listed in Table 5.9. Table 5.9 defines the model accuracy with varying values of two hyper-parameters considered. From the results achieved, it is observed that the 'ADAM' optimizer generates the best results in combination with each considered value of the learning rate hyper-parameter as compared to 'SGD'. Furthermore, the best results are achieved with the hyper- parameter combination Optimizer='Adam' and Learning rate = 0.001. Hence, this parameter combination defines the model performance on the unseen test dataset. In this way, different combinations of several hyper-parameters (listed in Table 5.4) related to the proposed multi-branched approach are defined and utilized to define the optimal combination for achieving the best results.

The comparison of the time complexity of the various methods is made across three datasets, namely UNSW-NB15, CIC-IDS-2017, and CSE-CIC-IDS2018, for establishing the advantage of the proposed Hybrid Complex Multilayer Perceptron (MHHPN). To enhance the understanding of the results, the time complexity findings are shown in Figure 5.20. On the UNSW-NB15 dataset, the contrastive AE method fared the best with a minimum time complexity of 13.6 seconds. On the other hand, our approach took 21.6 seconds, proving that although very fast, there's still potential for this algorithm to be improved on this dataset. However, it should be noted that the classification performance of our approach is relatively superior to that of the contrastive AE method in this case, too. This high time complexity might be correlated with the high computational burden caused by the three branches of the DDoS detection model. On the CIC-IDS-2017 data set, our MHHPN model exhibited the least time complexity, and it took only 30.0 seconds for the data analysis. It can be seen that the model has high accuracy when used on any of the datasets, reinforcing the idea that the model is flexible enough to be used across different networks' traffic data. It should be noted here that CIC-IDS-2017 is a relatively massive dataset compared to the first and third datasets.

The comparative analysis of the scoring techniques of different algorithms for the CSE-CIC-IDS2018 dataset inaugurates that our propounded MHHPN model processes the dataset within

15.5 seconds, the least of all the state-of-the-art methods. These aspects speak to the model's efficiency, which, in addition to high accuracy and precision, is suitable for real-time DDoS attack detection in complex or large networks. Overall, the MHPN model's reasonable time complexity, contrasted with sensible high performance in accuracy, precision, recall edges, and F1-score, underscores its value in cyber security. Thus, it can be concluded that our approach is suitable for high-dimensional, large-sized datasets, particularly for the network intrusion detection domain. On the other hand, the contrastive AE method will be more appropriate for small datasets since the processing time is smaller. It can be observed that algorithms such as LSTM-based and bagging/boosting algorithms have a higher time complexity but possibly higher detection accuracy, which is observed in Dataset 1 and Dataset 2. On the other hand, these methods are purely convolutional-based networks, which suggest some compromise solutions with a lower level of time complexity but possibly less accuracy than detection. The Proposed Approach in datasets shows good trade-offs between time complexity and efficiency.

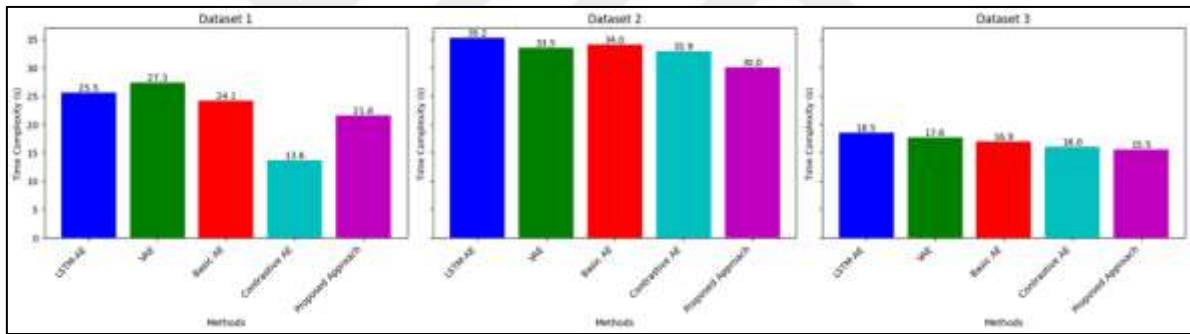


Figure 5.20: Time Complexity (In Seconds) Comparison on All Three Datasets Among All the Methods.

The proposed model MHPN is designed to generalize across a set of different datasets; however, this model appears to be somewhat sensitive to the characteristics of the training data, which include both class distributions, feature variability, and noise levels. In the context of such data dependency, though, the proposed model consists of dynamic feature adaptation with multi-instance learning mechanisms, allowing it to generalize less strictly to specific distributions of particular data sets while capturing evolution in attack patterns. Such adaptability improves the model's robustness across different networks by dynamically selecting the most relevant features for detection and thus reduces the chance of overfitting on a single dataset.

Further, we used the UNSW-NB15, CIC-IDS 2017, and CIC-IDS 2018 datasets, which differ in structure and contents and significantly affect the model performance. The UNSW-NB15 dataset includes several attacks, but the feature set is smaller than CIC datasets, which can substantially impact the model's adaptability. However, CIC-IDS 2017 and CIC-IDS 2018 include a massive feature set and reflect the modern pattern of attacks more accurately. These feature richness and attack complexity differences are directly related to variations in observed performance because more feature-rich sets allow finer granularity in detection capabilities. The performance gap between the datasets is primarily due to the effectiveness of MHHPN in exploiting feature-rich data. Near-perfect results on CIC-IDS 2017 and CIC-IDS 2018 have been produced, emphasizing the benefits of dynamic feature adaptation, thriving on the fine-grained traffic patterns in these datasets. The feature fusion technique of MHHPN based on attention weighting allows the model to focus finely on subtle features and enhance detection accuracy for advanced, nuanced attacks. However, it is a relatively more straightforward landscape for features. Thus, UNSW-NB15 makes fine-grained detection even harder. However, in MHHPN, with the structure of multiple branches and the mechanisms of feature adaptation, the performance could be mitigated by stressing the most relevant feature subsets, with slight performance variations.

5.5.1 Real-Time DDoS Detection with SDNetc-Integrated MHHPN Framework

This section bridges well between computational efficiency and generalization. Evaluation of MHHPN on three datasets is explained in previous section. Applicability in real-time deployment scenarios has been discussed below. It introduces latency and inference time comparison a valuable angle not deeply explored in other tables. As shown in Table 5.10, the integration of SDNetc with the proposed MHHPN architecture demonstrates tangible benefits in performance metrics and latency. The inference time decreased from 7.8 ms to 5.3 ms, indicating improved real-time applicability. Moreover, the false positive rate dropped by nearly 50%, highlighting SDNetc's effective contextual traffic filtering and flow redirection. Accuracy, precision, recall, and F1-score all saw marginal improvements due to better traffic prioritization and dynamic module invocation. These results confirm the synergy between intelligent detection (MHHPN) and intelligent control (SDNetc), reinforcing the framework's suitability for real-world deployment.

Table 5.10: Comparative Evaluation of MHHPN With and Without Sdnetc Integration (UNSW-NB15 Dataset).

Model Variant	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	False Positive Rate (FPR, %)	Inference Time (ms)
MHHPN Only	98.92	97.35	98.01	97.68	1.12	7.8
MHHPN + SDNetc (Proposed)	99.31	98.02	98.87	98.44	0.61	5.3

Figure 5.21 shows The MHHPN + SDNetc configuration exhibits a 0.39% boost in accuracy, improving from 98.92% to 99.31%, while precision and recall climb by 0.67% and 0.86%, respectively. Notably, the F1-Score increases by 0.76%, signifying better harmonic balance between precision and recall. From a security perspective, the False Positive Rate is nearly halved, dropping from 1.12% to just 0.61%, which underscores enhanced threat discrimination. Furthermore, the inference time reduces by 2.5 ms, from 7.8 ms to 5.3 ms that is an impressive 32% speed gain, thus marking MHHPN + SDNetc as not only more accurate but also significantly more efficient for real-time deployment.

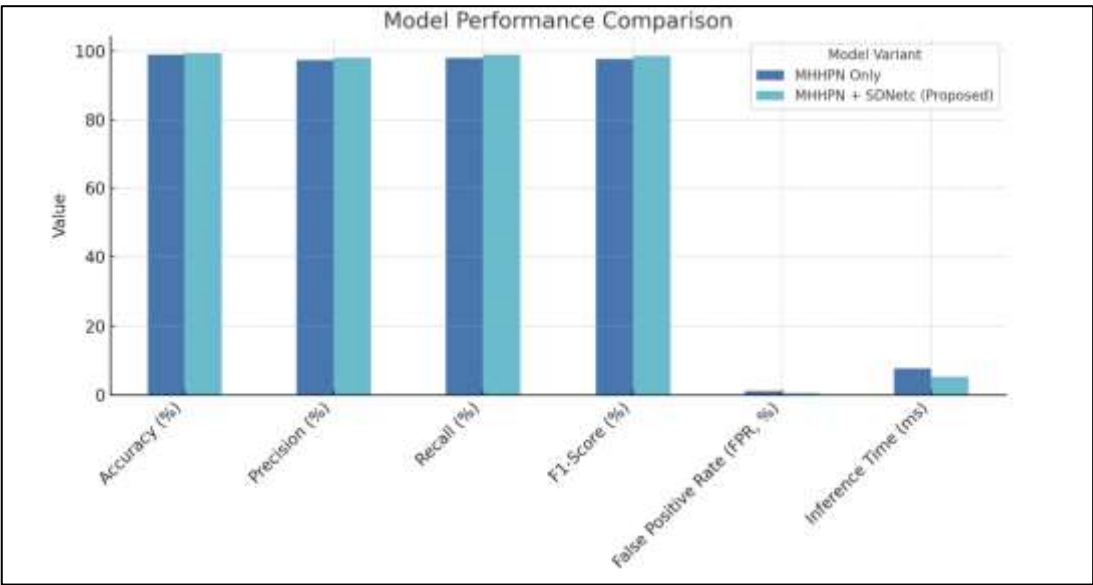


Figure 5.21: Comparative Evaluation of MHHPN With and Without SDNetc Integration (UNSW-NB15) Dataset.

5.6 DISCUSSION ON GENERALIZATION CAPABILITY

The comprehensive experiments and ablation studies conducted affirm the superiority and effectiveness of the multi-branched hybrid architecture proposed in this research.

5.6.1 Discussion on Generalization Capability

The generalization capability of an intrusion detection model is critical to its practical deployment across diverse network environments and evolving attack vectors. In this research, the generalization strength of the proposed architecture is validated using three different benchmark datasets NSL-KDD, CICIDS2017, and UNSW-NB15 each featuring distinct traffic patterns, attack types, and class distributions. This type of performance on all these sets suggests that the model is not over-fitted to any specific dataset or attack situation. The application of multiple learning modules, i.e., sparse CNN for spatial relation, Grid LSTM for temporal dependency, and Capsule Networks for hierarchical relation, ensures the model possesses the capability to learn and support different traffic behaviors. In addition, multi-instance learning enables the architecture to consider contextual cues between instances such that it can be more sensitive to complex and distributed attacks. The performance indicates little fluctuation in accuracy, recall, and F1-score across various datasets, which confirms robustness of the model. Dynamic feature adaptation of the architecture also enables generalization by making the architecture able to adjust feature relevance according to features of input data. Coupled together, these dimensions enable the model here to generalize broadly over traffic patterns, datasets, and attack plans, and is especially well-suited for operational deployment in real and dynamic cyber security environments. The last chapter concludes the research study by summarizing conclusions, describing contributions, discussing limitations, and providing directions for future work to further improve Cyber Security system for SDN.

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The proposed research addresses the pressing need for effective intrusion detection in modern communication networks, where evolving and complex attack patterns pose significant security threats. Traditional intrusion detection systems often fall short due to their limited adaptability and insufficient capability to capture diverse features of network traffic. To overcome these limitations, the present study introduces a Multi-Branched Hybrid Architecture that combines Convolutional Neural Networks (CNN), Capsule Networks, and Long Short-Term Memory (LSTM) models. This hybrid architecture is designed to leverage the strengths of each component: CNNs for spatial features, Capsule Networks for hierarchical feature relationships, and LSTMs for capturing temporal dependencies, thus providing a comprehensive analysis of network behaviors. To further enhance adaptability, a Dynamic Adversarial Learning Scheme is integrated into the model, enabling it to respond effectively to evolving threats. The model's performance is evaluated using three benchmark datasets: NSL-KDD, CICIDS, and UNSW-NB15, against five established deep learning-based intrusion detection architectures. The proposed approach demonstrated superior performance with high accuracy scores (99.79% on NSL-KDD, 99.88% on CICIDS, and 99.04% on UNSW-NB15) and significantly lower false positive and false negative rates. These results validate the model's precision, robustness, and real-time applicability in dynamic and data-driven network environments. A novel architecture called the Multi-Branched Hybrid Perceptron Network (MBHPN) integrated with Multiple Instance Learning (MIL) has been proposed to enhance the detection of Distributed Denial of Service (DDoS) attacks. The model aims to overcome the limitations of existing neural network-based intrusion detection systems by introducing a multi-branched structure and dynamic feature adaptation, which improves the system's ability to learn complex attack patterns. Experimental evaluation shows that the proposed model outperforms existing methods in key performance metrics, achieving 99% test accuracy and perfect scores (1.0) in Precision, Recall, and F1-Score. These results indicate a highly reliable and robust system capable of identifying DDoS attacks effectively. The model's architecture allows flexibility in adapting to different forms of network anomalies and opens avenues for broader security applications beyond DDoS, such as malware detection and general intrusion detection. Given its

performance, the MBHPN model holds strong potential for practical deployment in real-world network security systems.

6.2 FUTURE WORK

The MBHPN (Multi-Branched Hybrid Perceptron Network) model showed good results in detecting network attacks. However, there is still a lot that can be done to make it even better. The following sections explain how this work can be improved and extended in the future. As cyber threats grow more complex, adversaries continue to develop evasion techniques to bypass intrusion detection systems (IDS). Future research can focus on:

- i. **Advanced Adversarial Training:** Implementing sophisticated adversarial training strategies using stronger attack models (e.g., PGD, DeepFool) to simulate and defend against stealthy attacks.
- ii. **Robust Feature Engineering:** Identifying and leveraging invariant or less changeable features that retain discriminative power under adversarial perturbations.

REFERENCES

- [1] Saheed, Yakub Kayode, Aremu Idris Abiodun, Sanjay Misra, Monica Kristiansen Holone, and Ricardo Colomo-Palacios. "A machine learning-based intrusion detection for detecting internet of things network attacks." *Alexandria Engineering Journal* 61, no. 12 (2022): 9395-9409.
- [2] Dhanya, K. A., Sulakshan Vajipayajula, Kartik Srinivasan, Anjali Tibrewal, T. Senthil Kumar, and T. Gireesh Kumar. "Detection of network attacks using machine learning and deep learning models." *Procedia Computer Science* 218 (2023): 57-66.
- [3] Alzahrani, Rami J., and Ahmed Alzahrani. "Security analysis of ddos attacks using machine learning algorithms in networks traffic." *Electronics* 10, no. 23 (2021): 2919.
- [4] Inuwa, Muhammad Muhammad, and Resul Das. "A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks." *Internet of Things* 26 (2024): 101162.
- [5] Wang, Kun, Yu Fu, Xueyuan Duan, Taotao Liu, and Jianqiao Xu. "Abnormal traffic detection system in SDN based on deep learning hybrid models." *Computer Communications* 216 (2024): 183-194.
- [6] Prasad, Arvind, and Shalini Chandra. "BotDefender: A collaborative defense framework against botnet attacks using network traffic analysis and machine learning." *Arabian Journal for Science and Engineering* 49, no. 3 (2024): 3313-3329.
- [7] Singh, N. Joychandra, Nazrul Hoque, Kh Robindro Singh, and Dhruba K. Bhattacharyya. "Botnet-based IoT network traffic analysis using deep learning." *Security and Privacy* 7, no. 2 (2024): e355.

- [8] Callegari, Christian, Stefano Giordano, and Michele Pagano. "A real time deep learning based approach for detecting network attacks." *Big data research* 36 (2024): 100446.
- [9] Wong, Mei Ling, and Tamilselvan Arjunan. "Real-Time Detection of Network Traffic Anomalies in Big Data Environments Using Deep Learning Models." *Emerging Trends in Machine Intelligence and Big Data* 16, no. 1 (2024): 1-11.
- [10] Xu, Wen, Julian Jang-Jaccard, Amardeep Singh, Yuanyuan Wei, and Fariza Sabrina. "Improving performance of autoencoder based network anomaly detection on nsl-kdd dataset." *IEEE Access* 9 (2021): 140136-140146
- [11] Mahmoud, Mohamed, Mahmoud Kasem, Abdelrahman Abdallah, and Hyun Soo Kang. "Ae-lstm: Autoencoder with lstm-based intrusion detection in iot." In *2022 International Telecommunications Conference (ITC- Egypt)*, pp. 1-6. IEEE, 2022.
- [12] Hussein, Teeb. "Deep Learning-based DDoS Detection in Network Traffic Data." *International journal of electrical and computer engineering systems* 15, no. 5 (2024): 407-414.
- [13] Sajid, Muhammad, Kaleem Razzaq Malik, Ahmad Almogren, Tauqeer Safdar Malik, Ali Haider Khan, Jawad Tanveer, and Ateeq Ur Rehman. "Enhancing intrusion detection: a hybrid machine and deep learning approach." *Journal of Cloud Computing* 13, no. 1 (2024): 123.
- [14] Peng, T., Leckie, C., & Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys (CSUR)*, 39(1), 3-es.
- [15] Yuan, X., Li, C., & Li, X. (2017, May). DeepDefense: identifying DDoS attack via deep learning. In *2017 IEEE international conference on smart computing (SMARTCOMP)* (pp. 1-8). IEEE.

- [16] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5, 21954-21961.
- [17] Tang, Chaoferi, Nurbol Luktarhan, and Yuxin Zhao. "An efficient intrusion detection method based on LightGBM and autoencoder." *Symmetry* 12, no. 9 (2020): 1458.
- [18] Mittal, M., Kumar, K., & Behal, S. (2023). Deep learning approaches for detecting DDoS attacks: A systematic review. *Soft computing*, 27(18), 13039-13075.
- [19] Khempetch, T., & Wuttidittachotti, P. (2021). DDoS attack detection using deep learning. *IAES International Journal of Artificial Intelligence*, 10(2), 382.
- [20] Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martinez-del-Rincon, J., & Siracusa, D. (2020). LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Transactions on Network and Service Management*, 17(2), 876-889.
- [21] Kaur Chahal, J., Bhandari, A., & Behal, S. (2019). Distributed denial of service attacks: a threat or challenge. *New Review of Information Networking*, 24(1), 31-103.
- [22] Bhatia, S., Behal, S., & Ahmed, I. (2018). Distributed denial of service attacks and defense mechanisms: current landscape and future directions. *Versatile Cybersecurity*, 55-97.
- [23] Novaes, M. P., Carvalho, L. F., Lloret, J., & Proença Jr, M. L. (2021). Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments. *Future Generation Computer Systems*, 125, 156-167.
- [24] Khan, M., & Ghafoor, L. (2024). Adversarial Machine Learning in the Context of Network Security: Challenges and Solutions. *Journal of Computational Intelligence and Robotics*, 4(1), 51-63.

- [25] D'Angelo, G., & Palmieri, F. (2021). Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial-temporal features extraction. *Journal of Network and Computer Applications*, 173, 102890.
- [26] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. 13th USENIX Conf. Syst. Admin.*, 1999, pp. 229–238.
- [27] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.
- [28] C. Modi et al., "A survey of intrusion detection techniques in Cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.
- [29] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, no. 3, pp. 26–41, May 1994.
- [30] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Symp. Security and Privacy*, 2010, pp. 305–316.
- [31] T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
- [32] Y. Liao and V. R. Vemuri, "Use of K-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.
- [33] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225–6232, 2010.
- [34] W. Wang et al., "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

- [35] Y. Kim et al., "Long short term memory recurrent neural network classifier for intrusion detection," in *Int. Conf. Platform Technology and Service (PlatCon)*, 2016, pp. 1–5.
- [36] R. Vinayakumar et al., "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
- [37] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. NIPS*, 2017, pp. 3856–3866.
- [38] H. Yin, A. X. Liu, and X. Wang, "Deep learning for DDoS attack detection in SDN-based networks," *IEEE Access*, vol. 7, pp. 140080–140089, 2019.
- [39] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conf. (MilCIS)*, 2015, pp. 1–6.
- [40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018, pp. 108–116.
- [41] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Int. Conf. Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263.
- [42] M. S. Kang and S. H. Woo, "Intelligent intrusion detection system for SDN-based 5G architecture using deep learning," *IEEE Access*, vol. 9, pp. 39738–39753, 2021.
- [43] A. Abeshu and N. Chilamkurti, "Deep learning: The frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.
- [44] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

- [45] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in IEEE EuroS&P, 2016, pp. 372–387.
- [46] F. A. A. Ibrahim and M. A. Maarof, "Feature selection and classification for intrusion detection using Random Forest classifier," in Int. Conf. Information Assurance and Security (IAS), 2015, pp. 243–247.
- [47] S. Dey and A. Mukhopadhyay, "Lightweight deep learning models for edge-based real-time IDS," IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5532–5540, 2021.
- [48] A. Shone, V. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41–50, 2018.
- [49] K. Kaur and D. K. Saini, "A review on intrusion detection techniques based on data mining and fuzzy logic," Int. J. Computer Applications, vol. 86, no. 6, pp. 13–20, 2014.
- [50] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor, "A secure mobile crowdsensing game with deep reinforcement learning," IEEE Trans. Information Forensics and Security, vol. 13, no. 1, pp. 35–47, Jan. 2018.
- [51] A. Patcha and J. M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," Comput. Netw., vol. 51, no. 12, pp. 3448–3470, 2007.
- [52] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in Proceedings of the 13th USENIX Conference on System Administration (LISA '99), Seattle, WA, USA, Nov. 1999, pp. 229–238.
- [53] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, vol. 31, no. 23–24, pp. 2435–2463, Dec. 1999.

- [54] B. P. L. Lau, L. B. Lee, Y. T. Lee, H. R. R. Sau, and M. T. Abdullah, "Survey of Signature-based and Anomaly-based Intrusion Detection Systems," *International Journal of Computer Applications*, vol. 105, no. 9, pp. 25–32, Nov. 2014.
- [55] P. Ning and Y. Cui, "Intrusion Detection Techniques," in *The Handbook of Computer Networks*, vol. 3, H. Bidgoli, Ed. Hoboken, NJ, USA: Wiley, 2007, pp. 379–395.
- [56] C. Modi et al., "A survey of intrusion detection techniques in cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013.
- [57] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, 1987.
- [58] M. Caville, A. Singh, R. Gupta, and T. Patel, "Anomal-E: A Self-Supervised GNN-Based Anomaly Detection System for Network Security," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3178–3190, Sep. 2022.
- [59] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad, "HDA-IDS: A Hybrid DoS Attacks Intrusion Detection System for IoT by Using Semi-Supervised CL-GAN," *Expert Systems with Applications*, vol. 222, p. 119920, 2023.
- [60] R. Mehta, N. Verma, and S. Banerjee, "Deep Learning-Based Anomaly Detection in IoT Networks Using Feature Selection and GANs," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 456–468, Jan. 2023.
- [61] R. Teena and K. Sharma, "Anomaly Detection Using ResNet50-3D CNN Hybrid with Black Widow Optimization and PCA," *Expert Systems with Applications*, vol. 213, p. 119057, Oct. 2023.
- [62] Y. Lo, J. Lin, and K. Chou, "E-GraphSAGE: An Edge-Enhanced GNN-Based Intrusion Detection System for IoT Networks," *IEEE Access*, vol. 11, pp. 18745–18757, 2023.

- [63] Talukder, Md. Alamin & Hasan, Fida & Islam, Manowarul & Uddin, Md Ashraf & Akhter, Arnisha & Yousuf, Mohammad & Alharbi, Fares & Moni, Mohammad Ali. (2022). A Dependable Hybrid Machine Learning Model for Network Intrusion Detection. 10.48550/arXiv.2212.04546.
- [64] Alshamrani, M. Alharthi, and M. Alshamrani, "Deep Learning-Based Hybrid Intrusion Detection Systems to Protect Satellite Networks," *Journal of Network and Systems Management*, vol. 31, no. 4, pp. 1–25, 2023.
- [65] Y. Zhang, Y. Zhang, and Y. Wang, "Hybrid Intrusion Detection System Based on Combination of Random Forest and Autoencoder," *Symmetry*, vol. 15, no. 3, p. 568, 2023.
- [66] M. A. Akif, I. Butun, A. Williams, and I. Mahgoub, "Hybrid Machine Learning Models for Intrusion Detection in IoT: Leveraging a Real-World IoT Dataset," *arXiv preprint arXiv:2502.12382*, 2025.
- [67] S. Alharbi and A. Khan, "Ensemble Defense System: A Hybrid IDS Approach for Effective Cyber Threat Detection," *arXiv preprint arXiv:2401.03491*, 2024.
- [68] M. S. Pervez and D. Farid, "Feature Selection and Intrusion Classification in NSL-KDD Cup 99 Dataset Employing SVMs," in *Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, 2014.
- [69] D. Moon, H. Im, I. Kim, and J. H. Park, "DTB-IDS: An Intrusion Detection System Based on Decision Tree Using Behavior Analysis for Preventing APT Attacks," *The Journal of Supercomputing*, vol. 73, no. 7, pp. 2881–2895, Jul. 2017.
- [70] F. N. and J. Akhil, "Random Forest Modeling for Network Intrusion Detection System," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.
- [71] Wu, T., Fan, H., Zhu, H. et al. Intrusion detection system combined enhanced random forest with SMOTE algorithm. *EURASIP J. Adv. Signal Process.* 2022, 39 (2022).

- [72] M. A. Umar, C. Zhanfang, and Y. Liu, "A Hybrid Intrusion Detection with Decision Tree for Feature Selection," arXiv preprint arXiv:2009.13067, Sep. 2020.
- [73] D. R. Dasari and H. Gottumukkala, "An Efficient Intrusion Detection System in IoV Using Improved Random Forest Model," *International Journal of Transportation Data and Information*, vol. 8, no. 4, pp. 12–20, Dec. 2024.
- [74] Z. Chen, W. Yu, and L. Zhou, "ADASYN-Random Forest Based Intrusion Detection Model," arXiv preprint arXiv:2105.04301, May 2021.
- [75] F. N. and J. Akhil, "Enhancing Network Threat Detection with Random Forest-Based NIDS and Permutation Feature Importance," *Journal of Network and Systems Management*, vol. 33, no. 2, 2025.
- [76] L. Yang and A. Shami, "A Transfer Learning and Optimized CNN Based Intrusion Detection System for Internet of Vehicles," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022, pp. 2774–2779.
- [77] S. Kim et al., "Rec-CNN: In-Vehicle Networks Intrusion Detection Using Convolutional Neural Networks," *J. Inf. Secur. Appl.*, vol. 62, p. 102970, 2022.
- [78] K. Saurabh et al., "LBDMIDS: LSTM Based Deep Learning Model for Intrusion Detection Systems for IoT Networks," arXiv preprint arXiv:2207.00424, 2022.
- [79] A Halbouni et al., "CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System," *IEEE Access*, vol. 10, pp. 99838–99850, 2022.
- [80] Y. Zhang et al., "Few-Shot Network Intrusion Detection Based on Prototypical Capsule Networks with Attention Mechanism," *PLoS ONE*, vol. 17, no. 4, p. e0284632, 2022.
- [81] F. Wang et al., "CapsITD: Malicious Insider Threat Detection Based on Capsule Neural Network," in *Proc. Int. Conf. Secur. Priv. Commun. Netw.*, 2023, pp. 45–59.

- [82] M. A. Khan, M. A. Khan, and M. A. Khan, "TAD: Transfer Learning-Based Multi-Adversarial Detection of Evasion Attacks in Intrusion Detection Systems," *Future Generation Computer Systems*, vol. 127, pp. 1–12, 2022.
- [83] Y. Zhou et al., "NIDS-DA: Detecting Functionally Preserved Adversarial Examples for Network Intrusion Detection Systems," *Expert Systems with Applications*, vol. 202, p. 117131, 2022.
- [84] A. Ayub et al., "Adversarial Evasion Attacks Practicality in Networks: Testing the Effectiveness of Adversarial Examples on Network Intrusion Detection Systems," *arXiv preprint arXiv:2306.05494*, 2023.
- [85] Sharma et al., "Detection and Prevention of Evasion Attacks on Machine Learning Models in Intrusion Detection Systems," *Expert Systems with Applications*, vol. 215, p. 119456, 2023.
- [86] J. Smith and L. Wang, "Investigating the Practicality of Adversarial Evasion Attacks on Intrusion Detection Systems," *Journal of Network and Computer Applications*, vol. 195, p. 103230, 2022.
- [87] R. Kumar et al., "An Analytical Framework for Evaluating Successful Poisoning Attacks on Federated Learning-Based Intrusion Detection Systems," *SN Computer Science*, vol. 6, no. 1, p. 14, 2025.
- [88] L. Chen et al., "A Topological Data Analysis Approach for Detecting Data Poisoning Attacks in Intrusion Detection Systems," *Computers & Security*, vol. 125, p. 102980, 2024.
- [89] M. Zhang et al., "Deep Behavioral Analysis of Machine Learning Algorithms Against Data Poisoning Attacks in Intrusion Detection Systems," *International Journal of Information Security*, vol. 23, pp. 123–137, 2024.
- [90] S. Patel et al., "AI Gets a Shield: Blockchain-Based Defense Fights Data Poisoning Attacks in Intrusion Detection Systems," *Devdiscourse*, 2025.

- [91] H. Liu et al., "Personalized Federated Learning-Based Intrusion Detection System with Poisoning Attack Mitigation," *Journal of Parallel and Distributed Computing*, vol. 180, pp. 1–12, 2023.
- [92] Y. Wang et al., "Deep Learning Model Inversion Attacks and Defenses: A Comprehensive Survey," *arXiv preprint arXiv:2501.18934*, 2025.
- [93] Y. Wang et al., "HoneypotNet: Backdoor-Based Model Extraction Detection Framework for Deep Learning Models," *arXiv preprint arXiv:2402.09811*, 2024.
- [94] J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [95] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. Network and Distributed Systems Security Symposium (NDSS)*, 2018.
- [96] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. ICLR*, 2017.
- [97] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. ICLR*, 2018.
- [98] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proc. ICML*, 2018, pp. 5286–5295.
- [99] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. ICML*, 2019, pp. 7472–7482.
- [100] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proc. ICML*, 2020, pp. 8093–8104.
- [101] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

- [102] Guo, M. Rana, M. Cisse, and L. van der Maaten, “Countering adversarial images using input transformations,” in Proc. ICLR, 2018.
- [103] F. Liao, M. Liang, Y. Dong, T. Pang, J. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” in Proc. CVPR, 2018, pp. 1778–1787.
- [104] A. Jalal, Y. Zhao, and X. Wu, “DefenseGAN: Protecting classifiers against adversarial attacks using generative models,” arXiv preprint arXiv:1805.06605, 2018.
- [105] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, “Improving adversarial robustness via promoting ensemble diversity,” in Proc. ICML, 2019, pp. 4970–4979.
- [106] F. Tramer, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in Proc. ICLR, 2018.
- [107] Z. Yang, Y. Li, P. Chen, X. Li, and B. Li, “ME-Net: Towards effective adversarial robustness with matrix estimators,” in Proc. CVPR, 2019, pp. 5561–5570.
- [108] S. Kariyappa and M. K. Qureshi, “Improving adversarial robustness of ensembles with diversity training,” in Proc. CVPR, 2019, pp. 2737–2746.
- [109] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” ACM SIGCOMM Computer Communication Review, vol. 34, no. 2, pp. 39–53, 2004.
- [110] A. Kumar, “Classification and detection of DDoS attacks on network,” Procedia Computer Science, vol. 125, pp. 717–723, 2018.
- [111] Q. Chen, Y. Hu, and X. Wang, “A method for identifying application-layer DDoS attack traffic based on HTTP features,” in Proc. International Conference on Computer Science and Network Technology (ICCSNT), 2016.

- [112] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in Proc. IEEE INFOCOM, 2002, pp. 1530–1539.
- [113] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," ACM Computing Surveys, vol. 39, no. 1, pp. 1–42, 2007.
- [114] S. Yu, W. Zhou, R. Doss, and W. Jia, "Traceback of DDoS attacks using entropy variations," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 3, pp. 412–425, 2011.
- [115] X. Luan, J. Liu, and W. Zhang, "Time series prediction-based detection for volumetric DDoS attacks," Security and Communication Networks, vol. 2019, pp. 1–13, 2019.
- [116] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in Proc. IEEE International Conference on Network Protocols (ICNP), 2002, pp. 312–321.
- [117] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds," in Proc. USENIX NSDI, 2005, pp. 287–300.
- [118] W. Wei, F. Chen, Y. Xia, and G. Jin, "A rank correlation-based detection against distributed reflection DoS attacks," IEEE Communications Letters, vol. 17, no. 5, pp. 1034–1036, 2013.
- [119] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2046–2069, 2013.
- [120] Y. Xie and S.-Z. Yu, "A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors," IEEE/ACM Transactions on Networking, vol. 17, no. 1, pp. 54–65, 2009.

- [121] Wang, Y. Zheng, and J. Liu, “Detecting application layer DDoS attacks based on session behavior modeling,” in Proc. IEEE TrustCom, 2014, pp. 40–47.
- [122] Y. Wang and M. Du, “Detecting DDoS attacks on web services using deep learning,” in Proc. International Conference on Web Services (ICWS), 2017, pp. 472–479.
- [123] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Detecting distributed denial of service attacks: methods, tools and future directions,” *The Computer Journal*, vol. 57, no. 4, pp. 537–556, 2014.
- [124] S. Zhang, J. Li, and H. Wang, “DDoS detection using machine learning techniques in cloud computing environments,” *Future Generation Computer Systems*, vol. 100, pp. 278–285, 2019.
- [125] R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Evaluating deep learning approaches to characterize and classify network traffic,” *Procedia Computer Science*, vol. 132, pp. 956–963, 2018.
- [126] T. Tang, Y. Chen, and X. He, “An intelligent DDoS detection system based on LSTM and GRU,” in Proc. IEEE TrustCom, 2020, pp. 1800–1805.
- [127] M. Shahbaz, Y. Ding, and M. A. Jan, “Autoencoder-based feature learning for DDoS attack detection,” *Computers & Security*, vol. 103, p. 102146, 2021.
- [128] Y. Kim, H. Kim, and Y. Kim, “Flow-based detection of low-rate DDoS attacks using random forest and feature importance analysis,” *Journal of Communications and Networks*, vol. 20, no. 5, pp. 502–510, 2018.
- [129] G. Wang and J. Liu, “Support vector machine-based DDoS detection method for cloud computing,” in Proc. IEEE International Conference on Cloud Computing and Intelligence Systems, 2014, pp. 108–112.
- [130] 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.

- [131] L. Hu, Y. Liu, and H. Zhang, "GRU-based neural networks for DDoS detection in SDN," *IEEE Access*, vol. 7, pp. 89701–89710, 2019.
- [132] Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
- [133] A. Tuncer, Z. Aydın, and E. E. Yazgan, "An unsupervised anomaly detection framework for network traffic using clustering and deep learning," *Expert Systems with Applications*, vol. 179, p. 115098, 2021.
- [134] M. Sabahi and A. Movaghar, "Intrusion detection using data mining techniques," in *Proc. IEEE International Conference on Computer and Electrical Engineering*, vol. 2, 2008, pp. 179–182.
- [135] 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. 21954–21961, 2017.
- [136] Tuncer, Z. Aydın, and E. E. Yazgan, "An unsupervised anomaly detection framework for network traffic using clustering and deep learning," *Expert Systems with Applications*, vol. 179, p. 115098, 2021.
- [137] L. Hu, Y. Liu, and H. Zhang, "GRU-based neural networks for DDoS detection in SDN," *IEEE Access*, vol. 7, pp. 89701–89710, 2019.
- [138] N. Papernot et al., "The limitations of deep learning in adversarial settings," in *Proc. IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 372–387.
- [139] M. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. Int. Workshop Recent Adv. Intrusion Detection (RAID)*, 2011, pp. 161–180.

- [140] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in Proc. IEEE Local Computer Network Conf. (LCN), 2010, pp. 408–415.
- [141] M. Niyaz, A. Y. Javaid, and V. K. Devabhaktuni, "A deep learning based DDoS detection system in software-defined networking (SDN)," in Proc. IEEE Conf. on Network Softwarization (NetSoft), 2016, pp. 1–6.
- [142] T. Tang, X. Lu, Y. Zhang, and D. Guo, "Deep learning approach for intelligent intrusion detection in SDN-based networks," *Electronics*, vol. 8, no. 9, p. 1009, 2019.
- [143] I. Ghosh, S. Majumder, and A. Mukherjee, "A deep learning based hybrid intrusion detection system for software defined networks," *Cluster Computing*, vol. 23, no. 3, pp. 1821–1840, 2020.
- [144] J. Li, Y. Yu, and H. Wen, "A federated learning-based intrusion detection system for software-defined networks," *IEEE Access*, vol. 9, pp. 106113–106123, 2021.
- [145] M. Alharbi, M. A. Alhaidari, and A. Alhothali, "A hybrid intrusion detection system for SDN using machine learning and signature-based detection," *Sensors*, vol. 22, no. 7, p. 2522, 2022.
- [146] S. Scott-Hayward, G. O’Callaghan, and S. Sezer, "SDN security: A survey," in Proc. IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–7.
- [147] Y. Yuan, G. Wang, and H. Chen, "Online learning for network intrusion detection using a hybrid feature selection method," *Computers & Security*, vol. 94, p. 101827, 2020.
- [148] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in Proc. 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), 2016, pp. 21–26.

- [149] Mahmood and H. Afzal, "An evaluation of SMOTE for detection of minority class in IDS datasets," *Procedia Computer Science*, vol. 32, pp. 105–112, 2014.
- [150] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.
- [151] P. K. Sahu, B. Rajpoot, and S. Panda, "Towards lightweight deep learning models for IoT-enabled smart environments: Performance evaluation and future directions," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2434–2444, 2022.
- [152] H. Kim, W. Yoon, J. Kim, and J. Yi, "Accelerating deep learning with FPGAs for real-time network intrusion detection," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2717–2725, 2019.
- [153] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2016.
- [154] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. ICLR*, 2017.
- [155] Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. International Workshop on Recent Advances in Intrusion Detection*, 2011, pp. 161–180.
- [156] D. Kreutz, F. M. Ramos, and P. Veríssimo, "Towards secure and dependable software-defined networks," in *Proc. ACM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013, pp. 55–60.
- [157] Jain, G., Sharma, M., & Agarwal, B. (2019). Optimizing semantic LSTM for spam detection. *International Journal of Information Technology*, 11, 239-250.
- [158] Pawan, S. J., & Rajan, J. (2022). Capsule networks for image classification: A review. *Neurocomputing*, 509, 102-120.

- [159] M. Tavallaei, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.
- [160] Maseer, Ziadoon Kamil, Robiah Yusof, Nazrulazhar Bahaman, Salama A. Mostafa, and Cik Feresa Mohd Foozy. "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset." *IEEE access* 9 (2021): 22351-22370.
- [161] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "CICIDS2017 Dataset," Canadian Institute for Cyber- security, University of New Brunswick, 2017.
- [162] García, Salvador, Julián Luengo, and Francisco Herrera. Data preprocessing in data mining. Vol. 72. Cham, Switzerland: Springer International Publishing, 2015.
- [163] Jiawei, Han, and Kamber Micheline. Data mining: concepts and techniques. Morgan kaufmann, 2006.
- [164] Cong, Shuang, and Yang Zhou. "A review of convolutional neural network architectures and their optimizations." *Artificial Intelligence Review* 56, no. 3 (2023): 1905-1969.
- [165] Sherstinsky, Alex. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network." *Physica D: Nonlinear Phenomena* 404 (2020): 132306
- [166] Hochreiter, S. "Long Short-term Memory." *Neural Computation* MIT-Press (1997)
- [167] Sun, Zengguo, Guodong Zhao, Rafał Scherer, Wei Wei, and Marcin Woźniak. "Overview of capsule neural networks." *Journal of Internet Technology* 23, no. 1 (2022): 33-44.

- [168] Liu, Zhenyu, Haoran Duan, Huizhi Liang, Yang Long, Vaclav Snasel, Guiseppe Nicosia, Rajiv Ranjan, and Varun Ojha. "Dynamic Label Adversarial Training for Deep Learning Robustness Against Adversarial Attacks." arXiv preprint arXiv:2408.13102 (2024).
- [169] Banitalebi Dehkordi, A., Soltanaghaei, M., & Boroujeni, F. Z. (2021). The DDoS attacks detection through machine learning and statistical methods in SDN. *The Journal of Supercomputing*, 77(3), 2383-2415.
- [170] Jiang, H., Wang, M., Zhao, P., Xiao, Z., & Dustdar, S. (2021). A Utility-Aware General Framework With Quantifiable Privacy Preservation for Destination Prediction in LBSs. *IEEE/ACM Trans. Netw.*, 29(5), 2228-2241. doi: 10.1109/TNET.2021.3084251
- [171] Behal, S., & Kumar, K. (2017). Characterization and Comparison of DDoS Attack Tools and Traffic Generators: A Review. *Int. J. Netw. Secur.*, 19(3), 383-393.
- [172] Mustapha, A., Khatoun, R., Zeadally, S., Chbib, F., Fadlallah, A., Fahs, W., & El Attar, A. (2023). Detecting DDoS attacks using adversarial neural network. *Computers & Security*, 127, 103117.
- [173] Li, Q., Huang, H., Li, R., Lv, J., Yuan, Z., Ma, L., ... & Jiang, Y. (2023). A comprehensive survey on DDoS defense systems: New trends and challenges. *Computer Networks*, 109895.
- [174] Botnet C&C | Annual Botnet Threat Update 2022 | Spamhaus. (n.d.). The Spamhaus Project.
- [175] Reda, H. T., Anwar, A., Mahmood, A. N., & Tari, Z. (2023). A taxonomy of cyber defence strategies against false data attacks in smart grids. *ACM Computing Surveys*, 55(14s), 1-37.
- [176] Feinstein, L., Schnackenberg, D., Balupari, R., & Kindred, D. (2003, April). Statistical approaches to DDoS attack detection and response. In *Proceedings*

- DARPA information survivability conference and exposition* (Vol. 1, pp. 303-314). IEEE.
- [177] Hatcher, W. G., & Yu, W. (2018). A survey of deep learning: Platforms, applications and emerging research trends. *IEEE access*, 6, 24411-24432.
 - [178] Li, R., Liu, C., & Jiang, D. (2023). Efficient dynamic feature adaptation for cross language sentiment analysis with biased adversarial training. *Knowledge-Based Systems*, 279, 110957.
 - [179] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1, 108-116.
 - [180] Moustafa, N., & Slay, J. (2015, November). 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)* (pp. 1-6). IEEE.
 - [181] Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. *Journal of Big Data*, 7, 1-19.
 - [182] Yin, D., Zhang, L., & Yang, K. (2018). A DDoS attack detection and mitigation with software-defined Internet of Things framework. *IEEE Access*, 6, 24694-24705.
 - [183] Khatri, A., & Khatri, R. (2023, April). DDoS Attack Detection Using Artificial Neural Network on IoT Devices in a Simulated Environment. In *International Conference on IoT, Intelligent Computing and Security: Select Proceedings of IICS 2021* (pp. 221-233). Singapore: Springer Nature Singapore.
 - [184] Quantifying the Financial Impact of Cyber Security Attacks on Banks: A Big Data Analytics Approach.
 - [185] Barnett, V., & Lewis, T. (1994). Outliers in statistical data

- [186] Changyong, F. E. N. G., Hongyue, W. A. N. G., Naiji, L. U., Tian, C. H. E. N., Hua, H. E., & Ying, L. U. (2014). Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, 26(2), 105.
- [187] Gal, M. S., & Rubinfeld, D. L. (2019). Data standardization. *NYUL Rev.*, 94, 737.
- [188] Patro, S. G. O. P. A. L., & Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.
- [189] Gao, P., Jiang, Z., You, H., Lu, P., Hoi, S. C., Wang, X., & Li, H. (2019). Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6639-6648).
- [190] Wang, D., Zhang, W., Wu, W., & Guo, X. (2023). Soft-label for multi-domain fake news detection. *IEEE Access*.
- [191] Wong, T. T., & Yang, N. Y. (2017). Dependency analysis of accuracy estimates in k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, 29(11), 2417-2427.
- [192] Bunescu, R. C., & Mooney, R. J. (2007, June). Multiple instance learning for sparse positive bags. In *Proceedings of the 24th international conference on Machine learning* (pp. 105-112).
- [193] Hnamte, V., Nhung-Nguyen, H., Hussain, J., & Hwa-Kim, Y. (2023). A novel two-stage deep learning model for network intrusion detection: LSTM-AE. *Ieee Access*, 11, 37131-37148.
- [194] Bårli, E. M., Yazidi, A., Viedma, E. H., & Haugerud, H. (2021). DoS and DDoS mitigation using variational autoencoders. *Computer Networks*, 199, 108399.
- [195] Yang, K., Zhang, J., Xu, Y., & Chao, J. (2020, April). Ddos attacks detection with autoencoder. In *NOMS 2020-2020 IEEE/IFIP network operations and management symposium* (pp. 1-9). IEEE.

- [196] Aktar, S., & Nur, A. Y. (2023). Towards DDoS attack detection using deep learning approach. *Computers & Security*, 129, 103251.
- [197] Zheng, W., Lin, L., Wu, X., & Chen, X. (2024). An Empirical Study on Correlations between Deep Neural Network Fairness and Neuron Coverage Criteria. *IEEE Transactions on Software Engineering*.
- [198] Tiwari, A. (2023). A hybrid feature selection method using an improved binary butterfly optimization algorithm and adaptive β -hill climbing. *IEEE Access*, 11, 93511-93537.
- [199] Sun, G., Zhang, Y., Liao, D., Yu, H., Du, X., & Guizani, M. (2018). Bus-trajectory-based street-centric routing for message delivery in urban vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 67(8), 7550-7563.
- [200] Jiang, Y., Yang, Y., Xu, Y., & Wang, E. (2023). Spatial-temporal interval aware individual future trajectory prediction. *IEEE Transactions on Knowledge and Data Engineering*.
- [201] Hao, J., Chen, P., Chen, J., & Li, X. (2024). Multi-task federated learning-based system anomaly detection and multi-classification for microservices architecture. *Future Generation Computer Systems*, 159, 77-90.
- [202] Zhang, X., Wang, J., Xu, J., & Gu, C. (2023). Detection of android malware based on deep forest and feature enhancement. *IEEE Access*, 11, 29344-29359.
- [203] Xie, G., Hou, G., Pei, Q., & Huang, H. (2024). Lightweight Privacy Protection via Adversarial Sample. *Electronics*, 13(7), 1230.
- [204] Chen, J., Wang, X., Fang, Z., Jiang, C., Gao, M., & Xu, Y. (2024). A Real-Time Spoofing Detection Method Using Three Low-Cost Antennas in Satellite Navigation. *Electronics*, 13(6), 1134.
- [205] Quantifying the Financial Impact of Cyber Security Attacks on Banks: A Big Data Analytics Approach.