NETWORK INTRUSION DETECTION SYSTEM USING HYBRID DEEP LEARNING APPROACHES IN SOFTWARE DEFINED NETWORKING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ANKARA UNIVERSITY

by

Rachid BEN SAID

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY IN
COMPUTER ENGINEERING

ANKARA
2024

ABSTRACT

PhD Thesis

NETWORK INTRUSION DETECTION SYSTEM USING HYBRID DEEP LEARNING APPROACHES IN SOFTWARE DEFINED NETWORKING

Rachid BEN SAID

Ankara University
Graduate of the School of Natural and Applied Science
Department of Computer Engineering

Supervisor: Prof. Dr. İman ASKERBEYLİ

Network Intrusion Detection Systems (NIDS) have seen rapid development in both academia and industry, driven by the escalating cyber-attacks targeting governments and commercial entities worldwide. The recent development focuses on leveraging a new network architecture, namely, the Software-Defined Network (SDN), to implement NIDS with Deep Learning (DL) approaches to enhance network monitoring and security. The SDN is an emerging architecture that decouples the network control and forwarding planes. The network controller is programmable, supporting straightforward network policy enforcement and simplified network management. These features of SDN enable innovative applications, shaping a new networking paradigm capable of establishing NIDS. This research study introduces a two-stage approach. In the initial stage, a hybrid model combining Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) architectures is developed. In this approach, hybrid feature selection techniques employing the Random Forest (RF) classifier and Recursive Feature Elimination (RFE) are employed. Subsequently, an attention mechanism is incorporated into the hybrid model in the second stage, aimed at enhancing network intrusion detection for multiclass classification scenarios without using feature selection techniques. The efficacy of the proposed models is evaluated using widely recognized datasets such as UNSW-NB15 and NSL-KDD, alongside the specialized InSDN dataset tailored specifically for SDN environments. The results underscore the superior performance of the proposed model in achieving high accuracy compared to alternative models such as LeNet5, AlexNet, CNN, and CNN-LSTM.

February 2024, 93 pages

Key Words: NIDS, Deep Learning, Attention Mechanism, CNN, BiLSTM, Network Security

ÖZET

Doktora Tezi

YAZILIM TANIMLI AĞLARDA HİBRİT DERİN ÖĞRENME YAKLAŞIMLARI KULLANILARAK AĞ SALDIRI TESPİT SİSTEMİ

Rachid BEN SAID

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. İman ASKERBEYLİ

Küresel düzeyde hükümetlere ve ticari şirketlere yönelik artan siber saldırılara karşı akademide ve endüstride hızli gelişmeler yaşanmaktadır. Son zamanlarda ağ izleme ve güvenliğini artırmak amacıyla Derin Öğrenme (DL) yaklaşımlarıyla Ağ Saldırı Tespit Sistemleri- NIDS'ni uygulamak için Yazılım Tanımlı Ağlar (SDN) adı verilen yeni bir ağ mimarisinden yararlanılmaktadır. SDN, ağ kontrolü ve yönlendirme düzlemlerini birbirinden ayıran ve yeni ortaya çıkan bir ağ mimarisidir. Ağ kontrolörü programlanabilir olup, basit ağ politikası uygulamasını ve ağ yönetimini desteklemektedir. SDN özellikleri, NIDS'ni uygulayabilecek yeni bir ağ oluşturma paradigmasını belirleyerek kolaylaştırmaktadır.

Yazılım Tanımlı Ağ (SDN), tüm ağın tek bir yerden kontrol edilmesine ve yönetilmesine olanak sağlayarak ağ yönetimini basitleştirmek için tasarlanmıştır. SDN, günümüzün veri merkezi ağ altyapılarında yaygın olarak kullanılmaktadır, ancak Dağıtık Hizmet Engelleme (DDoS), web saldırısı ve Kullanıcıdan kök (U2R) saldırı gibi yeni tehdit biçimleri, SDN'ların yaygın olarak benimsenmesini kısıtlayabilecek önemli sorunlardır. Saldırganlar, değerli hedefler oldukları için SDN kontrolörlere çekicidir. Bir SDN kontrolörü bir saldırgan tarafından ele geçirilebilir ve trafiği kendi ihtiyaçlarına göre yönlendirmek için kullanılabilir ki bu da tüm ağ için yıkıcı sonuçlara yol açabilmektedir.

SDN ve derin öğrenme yöntemlerinin birleşik vizyonu, Ağ Saldırı Tespit Sistemi (NIDS) dağıtımının güvenliği için yeni olanaklar sunaraken, tespit modellerinin etkinliği ve eğitim veri setlerinin kalitesine bağlıdır. NIDS'ler için derin öğrenme son zamanlarda çeşitli konularda umut verici sonuçlar vermiş olsa da, çalışmaların çoğu veri fazlalığı ve dengesiz veri kümesinin etkisini göz ardı etmiştir. Sonuç olarak, bu anormallik tespit sisteminin esnekliğini olumsuz yönde etkileyerek optimum model performansının düşmesine neden olabilmektedir.

Bu çalışmada, ilk aşamada Evrişimsel Sinir Ağı (CNN) ve çift yönlü uzun kısa vadeli bellek (BiLSTM) modeli oluşturulmuştur; burada rastgele orman sınıflandırıcı (RF) ve özyinelemeli özellik eleme (RFE) ile hibrit özellik seçimi kullanılmıştır. İkinci aşamada ise çok sınıflı sınıflandırma için ağ saldırı tespitini geliştirmek üzere hibrit özellik seçimi olmadan hibrit modele dikkat mekanizması eklenmiştir. Önerilen modellerin etkinliği, en sık kullanılan veri kümesi (UNSW-NB15 ve NSL-KDD) kullanılarak test edilip değerlendirilmiştir. Ayrıca SDN'e özel olarak ayrılmış InSDN veri kümesi de kullanılmıştır. Sonuçlar, önerilen modelin LeNet5, Alexnet, CNN ve CNN-LSTM gibi diğer modellerle kıyaslamada yüksek doğruluk elde ettiğini göstermektedir.

Bulgular, ağ ortamlarında güçlü sızma tespiti için gelişmiş derin öğrenme mimarileri ve dikkat mekanizmalarının kullanılmasının önemini vurgulamaktadır. Bu çalışmada, yenilikçi metodoloji ve modeller kullanarak gelişmiş tespit yetenekleri sunulmaktadır ve böylece SDN ortamlarında ağ güvenliğini güçlendirerek NIDS'in ilerlemesine katkıda bulunmaktadır.

Bu çalışma aşağıdaki katkıları sağlanmaktadır;

- CNN-BiLSTM hibrit modeli, CNN ve BiLSTM mimarisini benimseyerek SDN'de NIDS gerçekleştirmek üzere geliştirilmiştir. CNN katmanı, ağ trafiği verilerinden yerel özellikler çıkarırken BiLSTM katmanı yerel özellikler arasındaki zamansal bağımlılıkları öğrenir.

- Veri setlerini dengelemek için rastgele aşırı örnekleme kullanıldı, ardından yüksek önemli özellikleri seçmek için bir RF sınıflandırıcı ve RFE algoritması kullanıldı, böylece CNN-BiLSTM önerilen model, bu özellikleri yüksek performansla eğitebildi.
- Dikkat mekanizmasını, özellik seçim teknikleri kullanmadan SDN'de NIDS'yi güçlendirmek için hibrit model CNN-BiLSTM ile birlikte kullandık.
- Önerilen modellerin etkinliği, üç ayrı referans veri kümesi olan UNSW-NB15, InSDN ve NSL-KDD kullanılarak doğrulandı.
- Önerilen hibrit derin öğrenme modellerinin etkililiği ve başarısı, doğruluk, duyarlılık, F1-score, kesinlik gibi bir dizi ölçüt uygulanarak test edildi. Bulgular, modelimizin çoğu değerlendirme metriği için CNN, AlexNet, LeNet5 ve CNN-LSTM modellerini geride bıraktığını göstermektedir.

Şubat 2024, 93 sayfa

Anahtar kelimeler: Ağ Saldırı Tespit Sistemleri, Derin Öğrenme, Dikkat Mekanizması, CNN, BiLSTM, Ağ Güvenliği

FOREWORD AND ACKNOWLEDGMENT

I want to express my sincere gratitude to those who have supported me in my pursuit of knowledge. I especially want to thank my supervisor, Prof. Dr. İman ASKERBEYLİ, for his consistent attention, ongoing guidance, and unwavering support throughout the entire research process. His invaluable help played a crucial role in helping me overcome the challenges of this study.

I am also deeply appreciative of the collaboration from other faculty members and the dedicated assistance from all the staff at our faculty during my time here.

I extend my profound thanks to my family, whose influence has emphasized the importance of education and who have been with me throughout this academic journey.

Rachid BEN SAID
Ankara, February 2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
BFA	Brute Force Attack
BILSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
DDoS	Distributed Denial of Service
DL	Deep Learning
DoS	Denial of Service
FN	False Negatives
FP	False Positives
IDS	Intrusion Detection System
IoT	Internet of Things
LSTM	Long Short-Term Memory
ML	Machine Learning
NIDPS	Network Intrusion Detection Prevention System
NIDS	Network Intrusion Detection System
R2L	Root to Local
RF	Random Forest
RFE	Recursive Feature Elimination
RNN	Recurrent Neural Network
SDN	Software Defined Networking
TN	True Negatives
TP	True Positives
U2R	User to Root

1. INTRODUCTION

1.1 Problem Statement

Since the advent of the digital age, cyber-attacks have been a persistent threat. However, as technology advances, the type, volume, motivations, and sophistication of cyber threats have seen a notable increase. Given the growing reliance of the economy on the Internet, organizations are compelled to embrace innovative technologies like big data, smart homes, smart farming, healthcare, IoT, cloud computing, and social media. There are varying predictions as to how many interconnected devices will exist in 2025, but most forecasts indicate that there will be up to 100 billion devices(Taylor, Baron, & Schmidt, 2015). A number of causes are driving this rise, including the global IoT revolution, the widespread usage of personal IoT devices like fitness trackers, and the release of a new generation of smart home gadgets that are enabled by ecosystems. This expansion, while fostering technological progress, concurrently elevates the susceptibility of businesses. Consequently, enterprises become attractive targets for malicious actors seeking to disrupt infrastructure, communications, and integrated supply chains, thereby resulting in tangible repercussions such as physical damage, financial losses, and erosion of credibility (Antonucci, 2017; Coburn et al., 2019).

SDN represents the future of networking by offering a breakthrough solution for the limitations of traditional network management. The key advantages of SDN lie in its openness, programmability, and flexibility. In contrast to traditional networking, SDN streamlines network management through two main approaches: Firstly, the SDN network is comprised of standardized switching hardware with uniform interfaces. Secondly, network control is centralized and delegated to a controller. SDN finds applications in various fields, including big data, smart farming, data centers, cloud computing, smart home, and IoT.

ANNs used in DL, a subset of ML, are modeled after the structure of neurons in the human brain. The word "deep" describes the way these layers are densely stacked; it is made up of several layers. NLP, image processing, and speech processing are just a few

of the fields in which DL specializes. With DL, more and more data can be used in the best possible way. It is challenging to use conventional data analysis methods to have a good insight into the data and extract meaningful information from it, and almost impossible to manage this massive data. Hence, DL focuses deep into the network to extract valuable, dependable, and accurate information. The emergence of high-performance data-processing hardware and the advancement of more robust DL algorithms are the two key factors that have significantly contributed to the integration of DL in cybersecurity operations(Sadiku, 2019).

The DL algorithms revive with a massive amount of data, making it possible to train it on an enormous security dataset. The NSL-KDD, UNSW-NB15, and InSDN datasets contain recent attacks and have many network flow features and records that make them suitable to train DL algorithms. Based on the above, the objective is to design, and test a Hybrid DL-based NIDS for SDN.

1.2 Objectives of the Thesis

In this research study, we examine the benefits of employing a unique DL model in combination with feature selection for the classification and detection of SDN attacks. As a result, the following are two hypotheses:

a) The goal of a NIDS using a hybrid CNN-BiLSTM DL model and hybrid feature selection is to improve attack detection and classification accuracy.
b) The inclusion of attention mechanisms in the proposed hybrid DL model improves its ability to identify and identify attacks.

In order to test these hypotheses, we have established the following objectives;

a) To demonstrate efficiency in SDN intrusion detection, a customized CNN-BiLSTM model with an RF classifier and RFE feature selection is established.

b) Examinate the performance of CNN-BiLSTM and apply it to diverse datasets to validate its effectiveness against other models like CNN, CNN-LSTM, AlexNet, and LeNet5.
c) Develop the Attention-based CNN-BiLSTM DL approach tailored for NIDS within SDN, ignoring the utilization of RFE techniques and RF classifier for feature selection.
d) To benchmark the Attention-based CNN-BiLSTM DL approach with CNN, CNN-LSTM, AlexNet, and LeNet5 models.

1.3 Contributions of the Research

An SDN was developed to streamline network management by enabling centralized control and oversight of the entire network from a single location. SDN has become a prevalent technology in modern data center networks. However, the emergence of new threats such as DDoS attacks, web attacks, and U2R attacks presents significant challenges that could impede the widespread adoption of SDNs. SDN controllers are particularly enticing targets for attackers due to their critical role. If compromised, an attacker can manipulate an SDN controller to route traffic according to their agenda, potentially leading to severe repercussions for the entire network. While the convergence of SDN and DL methodologies introduces new possibilities for enhancing IDS deployment security, the efficacy of detection models hinges on the effectiveness of the training datasets. This study aims to make distinct contributions in the following areas:

i. The CNN-BiLSTM hybrid model is developed to perform NIDS in SDN via the adoption the architecture of CNN and BiLSTM. The CNN layer extracts local features from the network traffic data and the BiLSTM layer learns the temporal dependencies among the local features.
ii. We balanced the datasets using random oversampling, after that we used an RF classifier and RFE algorithm to select the highly important features so

that the CNN-BiLSTM proposed model could train those features with high performance.

iii. We used the attention mechanism with the hybrid model CNN-BiLSTM to enhance the NIDS in SDN without using feature selection techniques.

iv. The effectiveness of the suggested models was verified by employing three distinct benchmark datasets: UNSW-NB15, InSDN, and NSL-KDD.

v. The efficacy and effectiveness of suggested hybrid deep learning models were tested by applying a set of measures including accuracy, recall, F1-score, and precision. The findings indicate that our model outperforms the CNN, AlexNet, LeNet5, and CNN-LSTM models for the majority of the evaluation metrics.

1.4 Scope of the Research

The study's scope encompasses the following elements:

i. The primary goal of the research is to employ the hybrid DL technique for NIDS in SDN.

ii. This study will include an analysis of the existing literature about NIDS in SDN. The review will specifically focus on computational intelligence methods that form the foundation of this research.

iii. The datasets employed in this research study consist of InSDN, UNSW-NB15, and NSL-KDD.

iv. This study is centered around a hybrid DL model, with the development of leverages CNN for efficient extraction of high-level local features and BiLSTM to enhance the contextual understanding available to deep learning algorithms.

v. The RF classifier and the RFE method for hybrid feature selection to increase the efficiency of the CNN-BiLSTM model in the first stage.

vi. In the second stage, the NIDS in SDN is enhanced by using the attention mechanism with the CNN and the BiLSTM without using hybrid feature selection.

vii. The efficacy and performance of the suggested deep learning algorithm were tested using a set of measures including F1-score, accuracy, recall, and precision. The findings indicate that our model performs better than CNN, AlexNet, LeNet5, and CNN-LSTM models.

1.5 The Organization

Chapter 1, Introduction: introduce the general information of research and its background. We discuss the prevailing issues within this research domain, outlining the problems, objectives, scope, and contributions of this research study.

Chapter 2, Background and Literature Review: provide an extensive review of relevant literature and information in related areas that lay the foundation for identifying the research problem and proposing solutions. It encompasses an overview of existing research, as well as insights and challenges pertaining to NIDS in SDN.

Chapter 3, In the first stage we provide details the research methodology and rationale for choosing the proposed approach in achieving the research objectives. We briefly outline the proposed approach, which includes the structure of the proposed CNN-BiLSTM model, hybrid feature selection, data pre-processing, and fine-tuning optimization of the model. In the second stage, the implementation of attention mechanism with the proposed CNN-BiLSTM model for NIDS in SDN: provide details of the technical steps involved in deploying the attention mechanism.

Chapter 4, Results, and discussion for both models are presented in this chapter Additionally, we evaluate the suggested model's performance in comparison to other DL models.

Chapter 5, Conclusion: In this chapter, we analyze and emphasize the research's contributions and discoveries. Additionally, we offer suggestions and recommendations for future research endeavors.

2. LITERATURE REVIEW AND BACKGROUND STUDY

2.1 Introduction

In recent years, the fusion of Software-Defined Networking (SDN) and Deep Learning has emerged as a promising paradigm for enhancing network security. This chapter focuses on the intersection of Network Intrusion Detection Systems (NIDS) within the framework of SDN, leveraging hybrid deep learning techniques. The preceding literature review lays the groundwork by examining the fundamental features of SDN, its architecture, and the evolving landscape of network security concerns and complexities. Additionally, it explores various deep learning approaches, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (BiLSTM), as well as architectures like AlexNet and LeNet5, alongside attention mechanisms.

Against this backdrop, the integration of NIDS into SDN environments represents a pivotal strategy in fortifying network defenses. By harnessing the capabilities of deep learning models, NIDS can effectively analyze network traffic patterns, and detect anomalies. However, this approach necessitates a nuanced understanding of both SDN architecture and deep learning methodologies to develop robust and scalable solutions. Thus, this chapter aims to explore the synergy between NIDS and SDN through the lens of hybrid deep learning, offering insights into novel techniques, evaluation metrics, and future research directions in bolstering network security infrastructure.

2.2 Literature Review

Advancing swiftly, the constant evolution of a diverse range of network-centric computing applications, each accompanied by intricate specifications, poses numerous fresh challenges to networking technologies. These applications, including but not limited to big data analysis, 5G, social networks, cloud computing, and mobile apps,

demand increased streamlined network management, bandwidth, dynamic network control, and more nimble software development processes. The emergence of cloud computing has intensified the expectations from computer networking, necessitating enhanced efficiency, versatility, and agility. The existing network's expansive control and management scope significantly hampers its ability to meet future service requirements. Moreover, the inherent characteristics of today's networks complicate the implementation and enforcement of consistent network-wide policies across heterogeneous network devices(Duan & Toy, 2016).

DL, a subset of ML, has proven effective across various research domains covering, but not restricted to image processing, NLP, and face recognition. Additionally, several methods of DL have been evaluated in NIDS research, in particular, CNN, BiLSTM, and attention mechanism models have been evaluated since they are capable of representing context, sequence, and time information in training data sets, respectively.

Elsayed et al. devised a hybrid intrusion detection system melding CNNs with LSTMs. This model is adept at extracting both spatial and temporal characteristics inherent in network traffic. Two regularization methods were utilized by the authors in order to reduce overfitting: L2 dropout and regularization. They utilized a recently published NIDS dataset, InSDN, within the context of SDN, as detailed by (M. S. Elsayed, N.-A. Le-Khac, & A. D. Jurcut, 2020). The proposed model notably enhances the detection performance against zero-day attacks, boasting a remarkable achievement with detection accuracy reaching 96.32%.

An NID technique combining hybrid sampling and deep hierarchical networks that is, CNN and BiLSTM was presented by Kaiyuan et al. In the first phase, they reduced the noise sample in the majority category with the use of the OSS and increasing the minority sample using SMOTE(Jiang, Wang, Wang, & Wu, 2020). With this hybrid technology, they balanced the data set to enable the model to acquire the features of minority samples and significantly reduce the model's training time. In the second phase, they used CNN to extract spatial characteristics and BiLSTM to extract temporal characteristics. In two datasets, NSL-KDD and UNSW-NB15, the suggested model

achieved high classification accuracy of 77.16% and 81.58%, respectively. Nevertheless, the NIDS in SDN was not included in this study.

However, there has been interest in integrating machine learning-based data preparation techniques including feature selection, augmentation, and reduction into SDN. In(Jacob & Wanjala, 2017), A large-scale experimental study using the NSL-KDD dataset was proposed as answer to the problems in the KDD Cup 99 dataset in order to get high accuracy of intrusion detection. Five effective machine learning techniques (Naïve Bayes, CART, SVM, RF, J48, and SVM) were used in the study. In order to reduce feature complexity, the correlation feature selection technique was used, and the NSL-KDD dataset has just 13 features.

In TSDL (Revathi & Malathi, 2013), A two-stage deep neural network model was built and proposed for NIDS, incorporating SoftMax as a classifier in the output layer together with a stacked auto-encoder. TSDL was designed and implemented to use multiclass classification to detect intrusions. Across several datasets, downsampling and a number of pre-processing methods have been applied to improve monitoring efficacy and detection rates. The accuracy of UNSW-NB15 detection was 89.13 %. The efficacy of the suggested model was primarily tested in this article using the UNSW-NB15 dataset; the NIDS in the SDN scenario were not taken into consideration.

The Authors of the study (Khan, Gumaei, Derhab, & Hussain, 2019) proposed various neural network models for NIDS. These models included seq2seq structures employing LSTM, variational auto-encoder, and fully connected networks. Several datasets, including UNSW-NB15, KYOTO-HONEYPOT, MAWILAB, and NSL-KDD, were used to assess how well the proposed approaches distinguished between malicious and legitimate packets in the network. In the preparation of the data for training and feature manipulation in neural networks, different pre-processing methods like normalization and one-hot encoding were applied. Enabling neural networks to extract complicated properties from the wide range of packet input is the main goal of these parameters.

The authors (Malaiya et al., 2018)) created a DNN architecture, utilizing four hidden layers, to identify intrusion attempts on the KDD cup99 dataset. To optimize data preparation and minimize data consumption, they implemented feature scaling and encoding techniques. Over 50 characteristics from diverse datasets were integrated into this endeavor. Consequently, advanced hardware GPUs were employed to manage the extensive array of characteristics, thereby reducing training time. Despite attaining high accuracy in their findings, it's important to note that the study did not primarily concentrate on NIDS within SDN.

In their work (Tang, Mhamdi, McLernon, Zaidi, & Ghogho, 2016)employed a DNN to identify flow-based anomalies in SDN. Their objective was to minimize the computational expense of attack detection by focusing on just six essential characteristics from the NSL-KDD dataset. The DNN model structure included 3 hidden layers, each with 3, 6, and 12 neurons, achieving a global accuracy of 75%. However, this accuracy fell below the threshold required for real-world implementations. Subsequently, the authors improved their model. by incorporating a GRU using the same NSL-KDD database in a later study (BOUKRIA & GUERROUMI, 2019). The upgraded model demonstrated a significant improvement, with a detection rate reaching 89%. Despite this improvement, it is worth noting that feature selection was not employed by the authors, a factor that could potentially enhance the results.

A threshold-optimized CNN-BiLSTM-ATT strategy was presented by the authors (Lan et al., 2020) to solve problems related to poor detection rates in minority classes and class imbalance. This approach connects a cutoff modification method based on the ROC curve with the CNN-BiLSTM-Attention model. Mississippi State University introduced an operational dataset in 2014, which was employed to evaluate the efficacy of this method. The outcomes demonstrated a remarkable 96.7% accuracy.

An IDS called Dense Attention-LSTM was introduced by (Cao et al., 2021) in their study. To identify temporal trends in network traffic data, this model makes use of an LSTM network that is built on a CuDNN. Moreover, it incorporates dense dilated

convolutions to reveal underlying properties of network traffic, an attention method to capture key aspects, and global max-pooling to compress data and improve generalization capabilities. The results of the performance evaluation showed that the suggested model effectively detected different assaults with an accuracy of 81.28% and obtained a binary classification accuracy of 92.65%.

Kaiyuan et al. suggested a NID method combining hybrid sampling with deep hierarchical networks i.e., CNN and BiLSTM. In the first stage, they used OSS to reduce the noise samples in the majority category and then increase the minority samples by SMOTE(Jiang et al., 2020). From this hybrid technique, they balanced the dataset so the model can learn features of minority samples and greatly reduce the model training time. In the second stage, they used CNN to extract spatial features and the BiLSTM to extract the temporal features. By using this method, the proposed model achieved good classification accuracy in two datasets UNSW-NB15 and NSL-KDD with 83.58% and 77.16%, respectively. But this work didn't deal with NIDS in SDN.

In their study (Yang et al., 2022), the authors employed an intrusion detection model featuring a multilayer attention mechanism tailored for a power information network. When comparing the performance of this model with and without the attention layer, it was observed that the inclusion of the attention layer led to a notable increase in the detection rate, specifically by 1.99%.

The authors (Ren, Yuan, Zhang, Shi, & Huang, 2023) proposed "CANET: A Hierarchical CNN-Attention Model for Network Intrusion Detection", a novel hierarchical CNN-Attention model, CANET, for network intrusion detection. CANET combines CNN and Attention mechanisms to extract local spatio-temporal features and uses Equalization Loss v2 (EQL v2) to address the class imbalance problem. The model outperforms state-of-the-art methods in terms of accuracy, detection rate, and false positive rate, achieving 89.39% accuracy, 98.93% detection rate, and 0.87% false positive rate on the UNSW-NB15 dataset and 99.77% accuracy, 99.72% detection rate, and 0.18% false positive rate on the NSL-KDD dataset.

In This work, the authors(Alsharaiah et al., 2024) discussed the development of a novel model for NIDS by combining LSTM with an attention layer for binary classification of network traffic data. The study aims to address limitations in current ML/DL models like SVM and KNN, which often lack accuracy due to restricted features. By utilizing the UNSW-NB15 dataset, which includes various network traffic behaviors, the model focuses on capturing spatial and temporal attributes using LSTM to capture enduring correlations in sequential data. The attention layer dynamically selects important features, enhancing the model's learning ability. The model is trained using optimization procedures like the Adam optimizer and evaluated based on metrics such as accuracy and confusion matrix analysis. Results from the experiments show that the model performs better than previous models, highlighting its effectiveness in enhancing NIDS precision and cybersecurity measures against dynamic threats with an accuracy rate of 92.6%.

The summarized and comparative information on various studies in this field is provided in Table 2.1.

Table 2.1 Several DL and ML approaches for NIDS in traditional networks and SDNs

Ref.	**Dataset**	**Method**	**Number of selected features**	**Type of attacks**	**The highest accuracy achieved**		**Type of network**
					Binary class classification	**Multi-class classification**	
(Bakhshi & Ghita, 2021)	NSL-KDD, UNSW-NB15 and CIC-IDS2017	CNN and GRU	-	Anomaly detection	93.10% on NSL-KDD, 91.21% on UNSW-NB15 90.17% on CIC-IDS2017	-	Traditional
(Tang et al., 2016)	Owned:800000+Packets	Neuro Evolution of Augmenting Topologies (NEAT)	3-packet-level features	Worm, DoS	-	90%	Traditional
(Prasath & Perumal, 2019)	NSL-KDD	Meta-Heuristic Bayesian Network Classification (MHBNC)	Extraction of features and pre-processing	DoS, U2R, Probe and R2L	Not reported	Not reported	Traditional
(Song et al., 2017)	KDD99	Decision Tree	10 features and 15 features	Anomaly or benign	82.48% for 10 features and 91.17% for 15 features	-	Traditional
(Dawoud, Shahristani, & Raun, 2018)	KDD-Cup 1999	Restricted Boltzmann Machine	41 features	General anomaly	Precision rate 94%	-	SDN
(Narayanadoss, Truong-Huu, Mohan, & Gurusamy, 2019)	Owned	ANN, LSTM, and CNN	3-flow-based features	Crossfire	87% for LSTM 80% for CNN	-	SDN

Table 2.1 Several DL and ML approaches for NIDS in traditional networks and SDNs (continue)

Ref.	Dataset	Method	Number of selected features	Type of attacks	The highest accuracy achieved		Type of network
(Alshamrani, Chowdhary, Pisharody, Lu, & Huang, 2017)	NSL-KDD	SMO, J48 and NB	41 features	DDoS, U2R, R2L, Probe	-	97%,98.76%,95.11%	SDN
(Tang et al., 2016)	NSL-KDD	Deep neural network (DNN)	6 features	DoS, R2L, U2R, and Probe	-	75.75%	SDN
(Mehdi, Khalid, & Khayam, 2011)	Real Traffic	Maximum entropy-TRW-Rate-limiting NETAD	-	DoS and Port scan	85%	-	SDN
(Sathya & Thangarajan, 2015)	NSL-KDD	Binary Bat algorithm (BBA) and Entropy	DDoS 27 features, Probe 33 features, R2L 32 features, U2R 26 features.	DDoS, Probe, R2L, U2R	-	91.10%	SDN
(Kokila, Selvi, & Govindarajan, 2014)	DARPA2000	SVM	-	DDoS	-	95.11%	SDN
(Niyaz, Sun, & Javaid, 2016)	Real Traffic	Auto-encoder	-	DDoS	99%	95.96%	SDN
(Mousavi & St-Hilaire, 2018)	Real Traffic	Entropy	-	DDoS	96%	-	SDN
(Le, Dinh, Le, & Tran, 2015)	DARPA 1999	C4.5Decision tree	-	DoS and Probe	-	96%	SDN
(Nanda, Zafari, DeCusatis, Wedaa, & Yang, 2016)	Real Traffic	DT, BN, NB,C4.5	-	Brute force	91.68%	-	SDN
(H. Li, Wei, & Hu, 2019)	NSL-KDD	RNN	-	Anomaly-based IDS	Not reported	Not reported	SDN
(Manso, Moura, & Serrão, 2019)	CAIDA	Rule-based	-	DDoS	-	-	SDN
(Albahar, 2019)	NSL-KDD, UNSW-NB15 and KDDCup 1999	RNN	-	Anomaly and benign	97.39% on NSL-KDD	-	SDN
This study	UNSW-NB, NSL-KDD, and InSDN	CNN-BiLSTM	10 features	BFA, U2R, DDoS, DoS, R2L, Botnet, Web Attack, Probe, and Normal	97.77 % on InSDN, 95.96% on NSL-KDD, 93.51 % on UNSW-NB15	97.12 % on InSDN, 98.42% on NSL-KDD, 84.23 % on UNSW-NB15	SDN

2.3 Software-Defined Network

2.3.1 Preface

Due to the COVID-19 pandemic and the necessary move towards remote working, online education, and entertainment content, there has been a significant global increase in dependency on the Internet since 2019. This unprecedented increase in internet usage has underscored the critical need for robust and scalable IT infrastructure solutions. Consequently, IT infrastructure providers are increasingly turning to cutting-edge technology like SDN to meet the escalating demands of this digital era. SDN offers a paradigm shift from traditional network architectures by centralizing network management and control, allowing for greater agility, scalability, and efficiency. Moreover, SDN's inherent flexibility enables rapid adaptation to evolving security threats and dynamic network environments, making it a compelling choice for organizations seeking to deal with the complex difficulties resulting from the pandemic-induced digital transformation.

Universally recognized as the original SDN deployment, OpenFlow research was disclosed in 2008 by a Stanford University study team. OpenFlow is founded on the notion of separate data and control planes with a single controller. A new networking architecture featuring distinct data and control planes as well as a centralized, programmable control platform was made possible by OpenFlow, which set off a wave of networking innovations. SDN is the name of this emerging networking architecture (Duan & Toy, 2016).

The word "Software-Defined" pertains to the capacity to encapsulate the administrative and operational functionalities of a particular technology. In the realm of computer networking, it involves governing the configuration of network devices, services (QoS), Traffic engineering, Virtual Local Area Networks (VLANs), and Firewall policies. To encapsulate, a system deemed as SD relies on software to attain Abstraction, Automation, and Adjustment characteristics. (Huang, Chowdhary, & Pisharody, 2018).

SDN tackles the inadequacy of conventional networks in meeting the intricate and scalable computing and storage requirements of contemporary applications. It achieves this by disentangling data transmission from network control, thereby enhancing network efficiency, programming, and management. The architectural comparison between traditional networks and SDN is illustrated in Figure 2.1 (Buyya & Son, 2018).

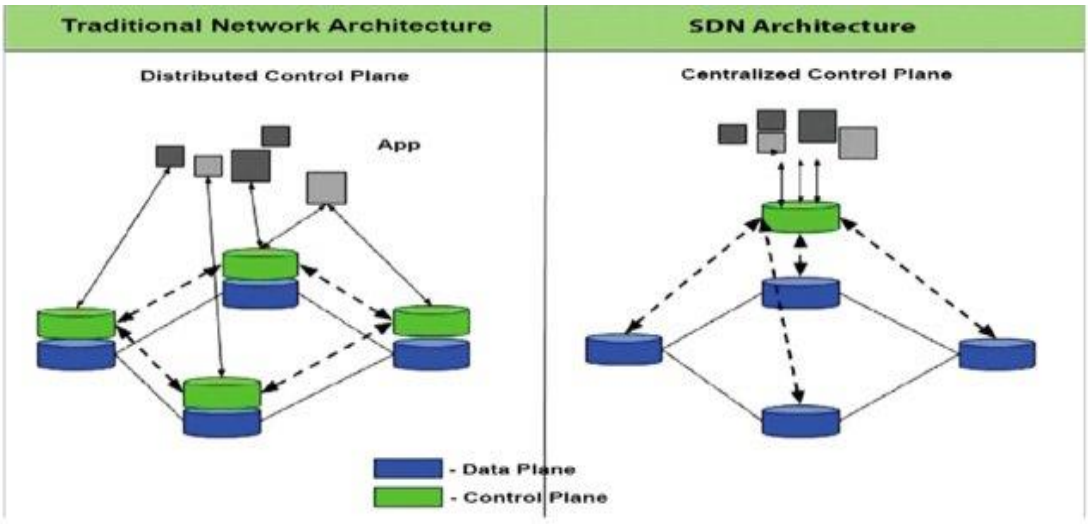

Figure 2.1 Comparison of SDN and Traditional Network architecture (Buyya & Son, 2018)

2.3.2 Fundamental features of SDN

Six fundamental characteristics of SDN are as follows:

- **Programmability**: One of the fundamental ideas of SDN is this. The controller's software applications can be used to configure network devices. By allowing the deployment of many network functionalities in software, SDN increases the network's programmability. It enables the network to completely adjust to the ever-changing needs of users, network managers, and applications (Nunes, Mendonca, Nguyen, Obraczka, & Turletti, 2014).
- **Centralization**: SDN replaces decentralized control architectures by centralizing all control in a single node, the network controller(Blenk, Basta, Reisslein, & Kellerer, 2015).

- **Abstraction**: a representation of an entity that only shows certain attributes while concealing others that are unrelated to the specifications. Network resource control is possible from a distance thanks to resource abstraction. Storage, processing, forwarding, and other functions needed to provide a service are examples of resources. Building scalable, adaptable, and modular network control systems is made possible by abstraction. Storage, forwarding, processing, and other functions needed to provide a service are examples of resources. Building scalable, adaptable, and modular network control systems is made possible by abstraction. Three basic abstractions could be present in an SDN.(Kreutz et al., 2014): forwarding, distribution, and specification.
- **Data and control plane isolation:** The splitting of the data and control planes enables network protocol experimentation to be conducted effectively. The timely transmission of control messages is made possible by the separated networking design (Nunes et al., 2014).
- **Network virtualization:** Several virtual networks can use a single physical network infrastructure due to network virtualization(Jain & Paul, 2013).
- **Openness**: Openness and interoperability stand as vital elements for SDN systems. SDN serves as an open platform, fostering creativity and connectivity. Because SDN is open-source, its interfaces should be open standards instead of commercial ones(Underdahl & Kinghorn, 2015).

SDN has shown great promise and is currently receiving a lot of attention in the following domains: data centers, LANs, wireless networks, WANs, Internet of Things.

2.3.3 Architecture of SDN

The evolution of computers is contrasted with the progression from closed, vertically integrated, and proprietary systems to an open computing framework, exemplified by the advancements in SDN. In SDN, the controller features a programmatic interface tailored for OpenFlow switches, akin to the hardware structure in computers. Through

this programmatic interface, network applications, referred to as net apps, can be developed to execute control and administration tasks while introducing novel functionalities. Notably, the control plane in SDN is logically centralized, and the design of network applications assumes a unified machine perspective, as depicted in Figure 2.2 (Azodolmolky, 2013)

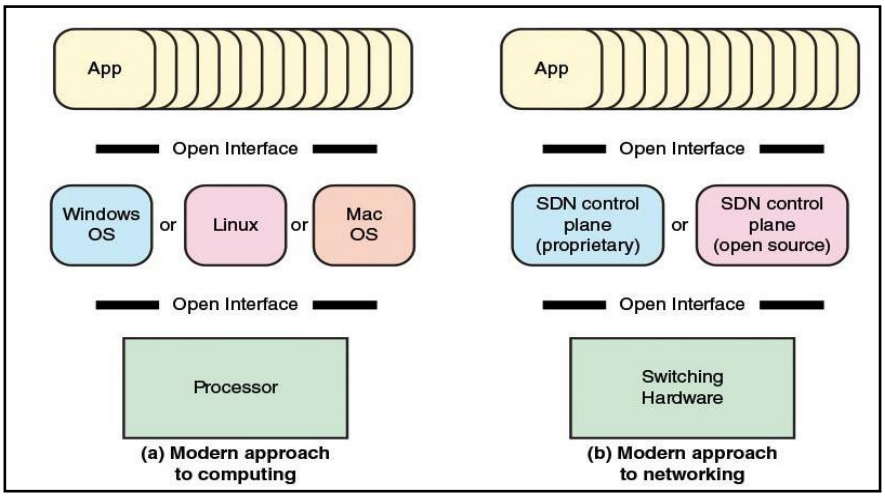

Figure 2.2 The current perspective on networking and computing (Stallings, 2015)

SDN represents a novel computer networking architecture that employs a structured API (Application Programming Interface). In conventional networks, essential network functions are embedded in hardware devices like switches, routers, and gateways. Introducing a new design typically requires updating all these hardware components. The existing traditional networking architectures exhibit scalability issues and limited functionality in meeting contemporary demands. SDN seeks to overcome these limitations inherent in current network infrastructures. It achieves simplification in two key ways: firstly, by employing uniform and standard interface switching equipment throughout the network, and secondly, by centralizing network control instead of distributing it. The fundamental SDN architecture is illustrated in Figure 2.3.

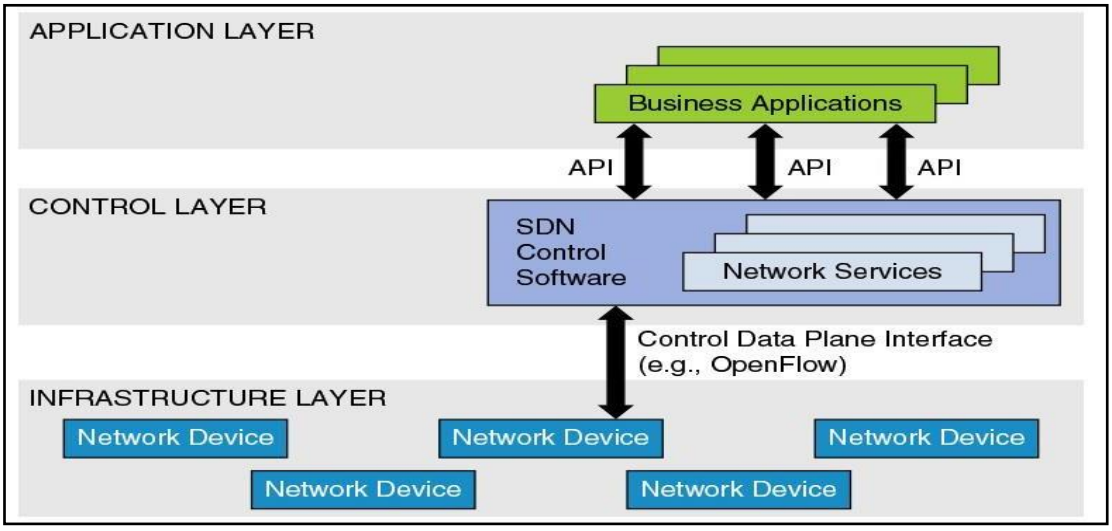

Figure 2.3 Fundamental SDN architecture (Doherty, 2016)

Three layers constitute SDN architecture: data, control, and application, with organized connections between the data and control layers (Hoang & Pham, 2015):

- **The application layer:** which is the top layer of the SDN architecture, is dedicated to several end-user applications. These applications utilize SDN communication and network services to fulfill user requirements. They encompass a range of functions such as routing, monitoring, security, load balancing, flow control, resource management, network virtualization, and traffic management.
- **Control layer:** The control layer oversees all decision-making protocols and access-control algorithms, taking charge of network control, routing decisions, and network programming. Interaction between controllers, applications, and devices is facilitated through APIs. Utilizing a global perspective, an SDN controller makes decisions that have global relevance. Programmability is One important aspect of the control layer, enabling the configuration of the network. In this paradigm, network devices are exclusively utilized for forwarding, with the controller responsible for determining the destination of data.
- **Data layer**: Known as the data layer, is comprised of specialized hardware

components such as switches, access points, and routers. Its primary role involves the processing of packets and the transmission of information provided by the upper controller. As such, packet forwarding is among the most significant jobs that the data plane.

SDN has four types of APIs: eastbound, westbound, southbound, and northbound. In addition to its architecture, SDN uses southbound APIs to send data from switches and routers downstairs. The main protocol for the southbound API interface is OpenFlow, which allows control messages to be sent back and forth between the controller and network devices. This protocol describes the communication methods that the controller and network devices use to communicate. On the other hand, northbound APIs in SDN establish connections with applications that manage network devices abstractly with support from the controller. Eastbound and westbound APIs serve as interfaces between multiple controllers, facilitating the sharing and synchronization of data.

2.3.4 SDN Security concerns and complexities

Sezer et al. covered a number of obstacles to the SDN's widespread adoption (Sezer et al., 2013). They took into account matters like efficiency, adaptability, security, and compatibility of the SDN with conventional networks. In(Kreutz, Ramos, & Verissimo, 2013), Kreutz et al. divided security risks in the SDN into seven vectors. Three unique vulnerabilities specific to SDNs include controller vulnerabilities, an absence of confidence between the apps and the controller., and a malicious attack targeting the administrator host. Using the STRIDE and attack tree approaches, Kloti et al. demonstrated security analysis and modeling techniques in the SDN (Klöti, Kotronis, & Smith, 2013) and found weaknesses in the SDN. A survey of SDN security implementation and SDN control plane security was published by (Scott-Hayward, O'Callaghan, & Sezer, 2013).

A couple of studies also addressed several of the SDN's security problems. Shin et al. developed a framework (S. W. Shin et al., 2013) for creating different security applications over the SDN. A feature of the framework allows for the resolution of

conflicts between the installation of flow rules by various network applications. As a result, it limits how certain non-security apps can get around firewall rules that have been put in place by security applications. Strong firewall applications were suggested by Hu et al. and Wang et al. in (Hu, Han, Ahn, & Zhao, 2014) and (J. Wang et al., 2013), respectively, using the ow and firewall authorization space checks. For rule-conflict identification, they took into account the intra-table rule dependencies inside the switches' own tables. Shin and others. To decrease the data plane and control plane's interaction during adversarial harm, an expansion to the OpenFlow (OF) data plane was proposed (Shin, Yegneswaran, Porras, & Gu, 2013). It decreases the controller's burden of managing numerous ow requests from the switches. Nevertheless, the enhancement is restricted to TCP-based flows alone. Liyanage et al. suggested a HIP-based secure control channel architecture for mobile SDN (Liyanage, Ylianttila, & Gurtov, 2014). The design to protect the SDN controller from a malevolent administrator was proposed by Matsumoto et al. (Matsumoto, Hitz, & Perrig, 2014). In(Wen, Chen, Hu, Shi, & Wang, 2013), Wen et al. suggested a fine-grained permission scheme for controlling network applications' access on the controller and controlled network.

2.3.5 NIDS in SDN

IDSs primarily fall into two different categories: signature-based IDSs and anomaly-based IDSs(Fellin & Haney, 2014). Signature-based IDSs utilize pattern-matching techniques, where predefined signatures define abnormal behaviors in the network. However, a limitation of signature-based IDSs is their inadequacy in detecting novel attack types. However, anomaly-based IDSs model the normal behavior in the network and employ deep learning techniques to identify exceptions. Anomaly-based IDS have the capability to identify novel attack types, rendering them more powerful compared to signature-based IDSs in this regard(Amudha & Rauf, 2011).

Each component in the NIDS structure and its purpose are discussed and explained in this section. The NIDS is designed as the SDN controller. The goal of this study is to show the usage of the SDN architecture as a network infrastructure for NIDS. Figure

2.4 depicts the architecture of NIDS in SDN, which is composed of three major components: Anomaly Mitigator, Anomaly Detector, and Flow Collector.

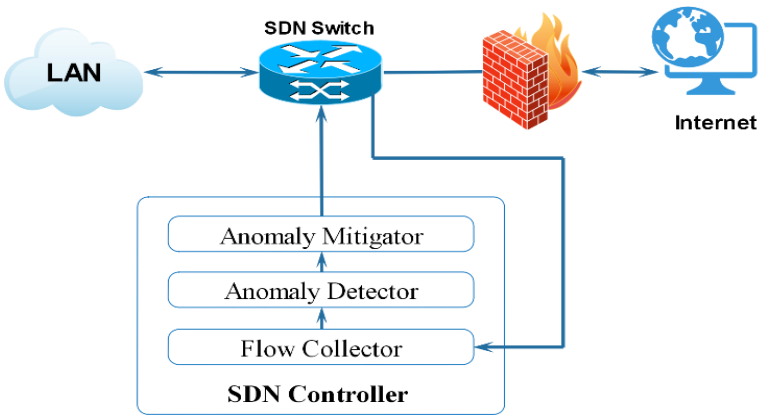

Figure 2.4 NIDS Architecture in the SDN controller

The Anomaly Mitigator can make flow choices based on the results of the Anomaly Detector (e.g., forward or drop the flow). The Anomaly Detector runs a trained model, gets network information, and determines whether or not a flow is an anomaly. A message or timer function triggers the flow collector, which aggregates all flow statistics, including the destination and source ports, and the destination and source IP protocol. All aggregated characteristics will be obtained by the Anomaly Detector module.

Figure 2.5 depicts the architecture of NIDS in SDN using the SDN architecture, which consists of three major components:

- The application layer's primary responsibility is to handle all network management activities that may be executed with the assistance of an NIDS and an SDN controller.
- The control layer manages data and traffic to start and stop each network flow.
- The infrastructure layer comprises two major components: hardware and software, such as routers and OpenFlow switches.

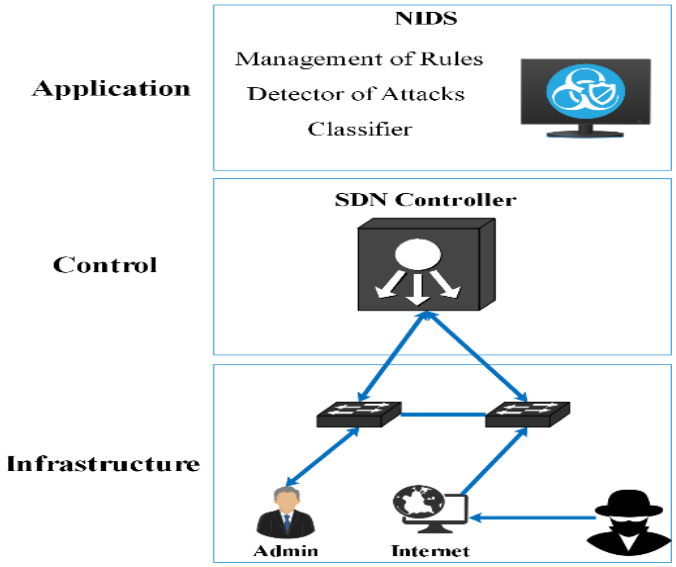

Figure 2.5 The architecture of NIDS in SDN

2.4 Deep Learning

2.4.1 Deep learning approaches

We are fortunate to live in a time of exceptional technological advancement and possess a vast amount of data, both organized and unorganized. This wealth of information has become the cornerstone of modern AI, which has its roots in the evolution of ML during the latter half of the 20th century. ML, as a subfield of AI, encompasses a suite of algorithms designed for autonomous learning, wherein knowledge is extracted from data to facilitate predictive analytics. Gone are the days of manual rule derivation and model construction by human hands to tackle vast datasets. Instead, ML heralds a paradigm shift in computer science, enabling systems to imbibe knowledge autonomously. As succinctly put by Arthur Samuel in 1959, ML can be broadly defined as the discipline that empowers computers with the capacity to learn without explicit programming(Géron, 2019). This transformative capability underscores the essence of ML, shaping the landscape of contemporary technology and paving the way for unprecedented advancements.

2.4.1.1 Convolution neural network

In DL, a CNN or ConvNet constitutes a class DNN primarily employed in computer vision studies. CNNs typically comprise one or more convolutional layers followed by one or more fully FC. These architectures are tailored to process two-dimensional input data efficiently. Local connections with assigned weights, coupled with various pooling layer configurations, yield translation-invariant features. Furthermore, CNNs have a great deal fewer parameters and simpler training than fully connected networks with comparable hidden units.(LeCun et al., 1989).

CNNs diverge notably from other machine learning algorithms by integrating both feature extraction and classification stages. The basic CNN architecture comprises five distinct layers: an input layer, a convolutional (CONV) layer, a pooling layer, a fully connected (FC) layer, and an output layer. These layers are segmented into two phases: feature extraction and classification. The feature extraction phase encompasses the input layer, CONV layer, and pooling layer, while the classification phase involves the FC layer and output layer.

The input layer defines fixed-size input data, which is then processed by the convolutional layer using learned kernels with shared weights. Subsequently, the pooling layer reduces the data size while providing the ability to retain pertinent information. Feature maps are the results of the feature extraction step. These retrieved characteristics are included in the FC layers during the classification process. Finally, one output neuron for each type of object makes up the output layer.

Although convolutional layers (CNNs) were designed for multi-dimensional data, images may be employed for intrusion detection tasks using one-dimensional versions of CNNs.

2.4.1.2 Recurrent neural network

RNNs are specifically crafted to handle sequential data. This type of data is pervasive across numerous applications, including language translation, speech recognition, and video recognition. Sequential data in language translation problems is translated sentence after sentence from one language to another. Speech recognition tasks involve converting audio clips into text, representing sequential data in the form of spoken words. Similarly, in video activity recognition, sequential data characterizes the activity depicted in a video by transforming sequential video frames into textual descriptions. RNNs are extremely useful tools in a variety of applications requiring sequential information processing because they are excellent at capturing the temporal dependencies found in such sequential data.

At the core of RNNs lies their unique ability to effectively manage temporal dynamics when analyzing elements within a sequence. Unlike other neural network architectures, RNNs are inherently designed to incorporate time into their processing mechanism. Consequently, the outputs generated by RNNs are affected by more than just the current input but also by the previous hidden states of the network. This makes it possible to maintain an internal memory of past inputs as they sequentially process each element, such as words within a sentence. As a result, when predicting the next word in a sentence, RNNs retain information about all preceding words and their interconnections, enabling them to capture and leverage the sequential nature of the data effectively.

In Figure 2.6, we observe the structure of an RNN featuring a loop that serves to retain sequential data. This diagram provides an expanded view, illustrating multiple instances of RNNs interconnected together. RNNs communicate by transmitting information from one state to the subsequent one. Consequently, an RNN with a loop signifies the presence of numerous RNN copies within the network architecture. An RNN uses the current input in addition to the prior state value to calculate its current state. Different versions of RNNs that are often encountered are GRU, LSTM, bi-directional RNNs, and

deep RNNs. Each of these variants has specific functions and uses in the context of sequential data processing.

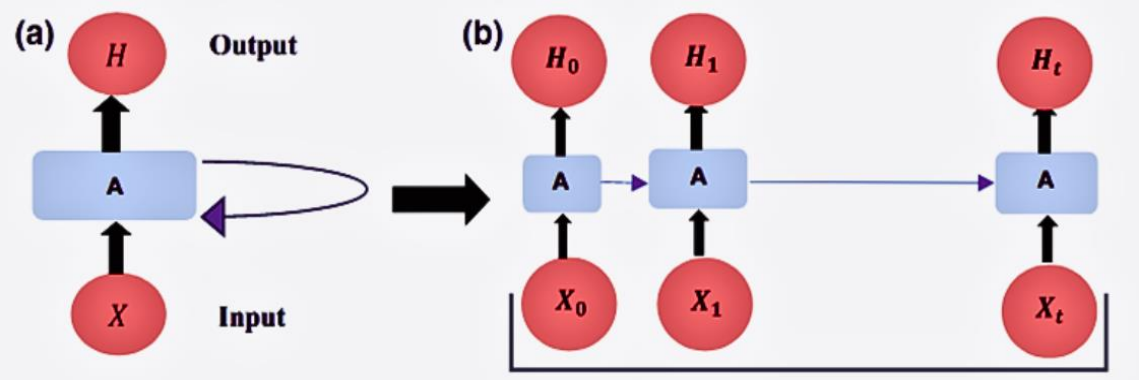

Figure 2.6 An RNN model with integrated information from preceding stages, alongside an unrolled representation of the RNN showcasing all previous states

2.4.1.3 Long short-term memory

The RNN is a sort of NN that links loops, adding feedback and memory over time. This memory helps this network type to learn and generalize in input sequences rather than individual patterns. An advanced RNN type called the LSTM Network was particularly useful when stacked in deep configuration(Brownlee, 2016).

LSTM revolutionizes the architecture of RNN by introducing a novel structure that incorporates gates and a memory unit, thereby redefining the traditional layer configuration of RNN. LSTM aims to store data in the memory cell for later use and upgrades. By employing this redesigned architecture, LSTM effectively addresses the challenges posed by gradient explosion and vanishing problems encountered in conventional RNNs. Furthermore, LSTM holds great promise for addressing network security challenges due to its variants' ability to capture both long and short-term dependencies effectively.

Figure 2.7 displays LSTM architecture, which consists of three crucial components: the forget gate x(t), the input gate i(t), and the output gate o(t). The forget gate f(t) plays the

vital role in determining which information should be discarded from the cell at time t. It achieves this by considering the both previous and current state of information; y(t−1) and x(t) respectively, ultimately producing a result of 0 indicating complete discarding, or 1 indicating complete retention.

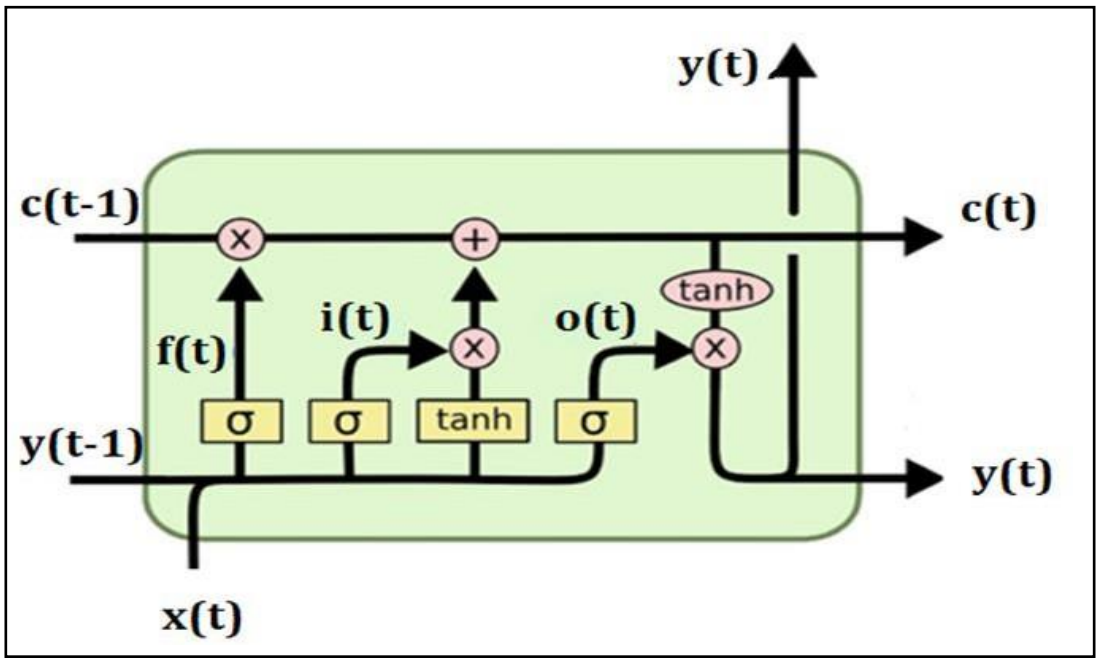

Figure 2.7 LSTM architecture (Bonaccorso, 2020)

The input gate decides what information will be passed on to the next cell. The input gate is defined as:

$$i(t) = \sigma(Wi \times [y(t-1), x(t)] + bi) \quad (2.1)$$

A memory unit is a part of the LSTM algorithm. Which of the prior memory contents is to be lost is determined by the forgetting gate. The definition of it is as follows:

$$f(t) = \sigma\,(Wf \times [y(t-1), x(t)] + bf) \quad (2.2)$$

The control is used for updates in the cell and is defined as follows:

$$\hat{C}(t) = \tanh(Wc \times [y(t-1), x(t)] + bc)$$
$$C(t) = f(t) * C(t-1) + i(t) * \hat{C}(t) \quad (2.3)$$

Hidden layer y(t—1) is updated by the output layer and is defined as:

$$o(t) = \sigma(Wo \times [y(t-1), x(t)] + bo)$$
$$y(t) = o(t) * \tanh(C(t)) \quad (2.4)$$

The sigmoid activation function is defined by the symbol in the equations, the appropriate weight matrices are denoted by w, and values are scaled from -1 to +-1 using tanh(Tai, Socher, & Manning, 2015). The sigmoid process is used to determine whether data from the preceding cells and data entering the subsequent cell will be forgotten (Sundermeyer, Schlüter, & Ney, 2012).

2.4.1.4 Bidirectional long short-term memory

Bi-LSTM is one of the RNNs, that aims to address the limitations of LSTM which exists in handling text sequence features. It excels in sequential modeling tasks compared to LSTM, as noted in studies by Niu et al. (2017) and Liu et al. (2017).

The weakness of LSTM is that information flows in only 1 direction, typically from the past to the future. In contrast, Bi-LSTM overcomes this constraint by allowing information processing in both directions simultaneously, leveraging two hidden states to capture forward and backward dependencies effectively. This structural feature positions Bi-LSTM as a pioneering choice in sentiment classification, as it is more effective in learning contextual information. You can observe the architecture of Bi-LSTM in Figure 2.8 (Schuster and Paliwal, 1997). By incorporating information from both preceding and succeeding sequences, Bi-LSTM keeps input data, unlike the standard RNN which needs decay in order to include future data.

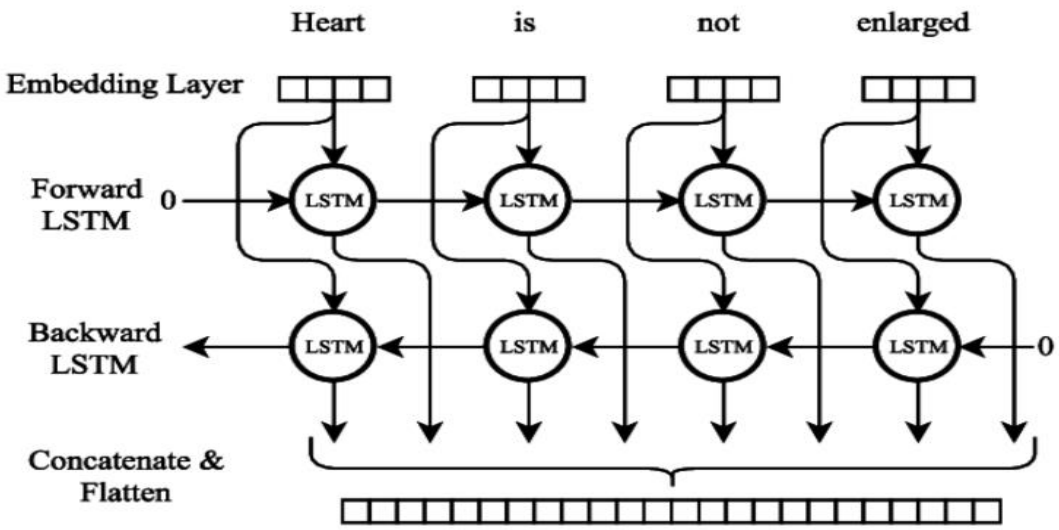

Figure 2.8 Bi-LSTM architecture (Schuster and Paliwal, 1997)

2.4.1.5 AlexNet

The AlexNet model (Wencheng, Xiaopeng, Hong, & Limin, 2018; Zhou & Wang, 2020), introduced by Krizhevsky and collaborators in 2012, secured the top position in the ILSVRC competition of that year. Its architecture comprises 8 layers, including 5 convolutional layers and 3 fully connected layers. The first, second, and fifth convolutional layers are followed by a pooling layer, specifically maximum pooling, while the third and fourth convolutional layers lack a pooling layer. Each convolutional layer utilizes the ReLU activation function and local area Normalized LRN. The final three fully connected layers convert the two-dimensional feature maps into one-dimensional column vectors. To prevent overfitting, the sixth and seventh fully connected layers continue to use the ReLU function as the activation function, while the eighth layer is fully connected. Serving as the classification layer, the connection layer does not include an activation function(T. Wang et al., 2020; Zhou & Wang, 2020).

2.4.1.6 LeNet5

The LeNet5 model is a CNN introduced by Professor Yann LeCun in the paper named "Gradient-based learning applied to document recognition" in 1998(LeCun, Bottou, Bengio, & Haffner, 1998). LeNet5 stands as a timeless example of a CNN, a type of

feed-forward NN. These networks possess artificial neurons capable of responding to specific portions within their coverage range, making them particularly effective for extensive image processing. LeNet5 employs convolutional operations, pooling mechanisms, and fully connected layers. Initially applied to the MNIST dataset for handwritten digit recognition, its architectural blueprint has influenced subsequent networks like AlexNet and VGG.

The LeNet5 CNN design has 7 layers total: 2 fully connected layers, 2 subsampling levels, and 3 convolutional layers. Figure 2.9 below shows a depiction of the LeNet5 architecture. The initial layer serves as the input layer, though it is typically not regarded as a learning layer. Designed to receive input images of 32x32 dimensions, this layer acts as the gateway for subsequent processing. In LeNet5, the convolutional layer serves as the initial hidden layer, where every multilayer kernel has a size of 5 x 5. After that, the convolutional feature map from the previous layer is compressed with the aid of the second layer, which serves as a pooling layer. With 16 sets of filters, the third hidden layer is also another convolutional layer. Subsequently, the 4th hidden layer is a pooling layer, further compressing the data. The 5th layer takes the form of a fully linked layer including 3 complete connections, ultimately organized by a classifier. Parameters that can be effectively changed in real-world applications include the size and number of the Conv and pooling layers.

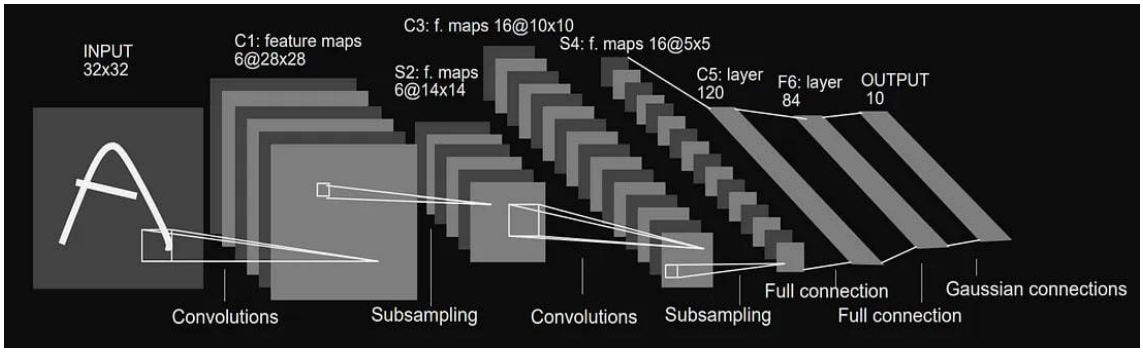

Figure 2.9 Architecture of LeNet5 model (Guan et al., 2020)

2.4.1.7 Attention mechanism

To help neural machine translation retain long source phrases, Bahdanau introduced the attention mechanism(Denning, 1987). The Attention mechanism creates links between the full source input and context vector, in contrast to the traditional way of creating a single context vector. These connections, referred to as shortcuts, feature adjustable weights for each output feature (as depicted in Figure 2.10). This approach accentuates the significance of significant parts of the input data while diminishing the importance of others.

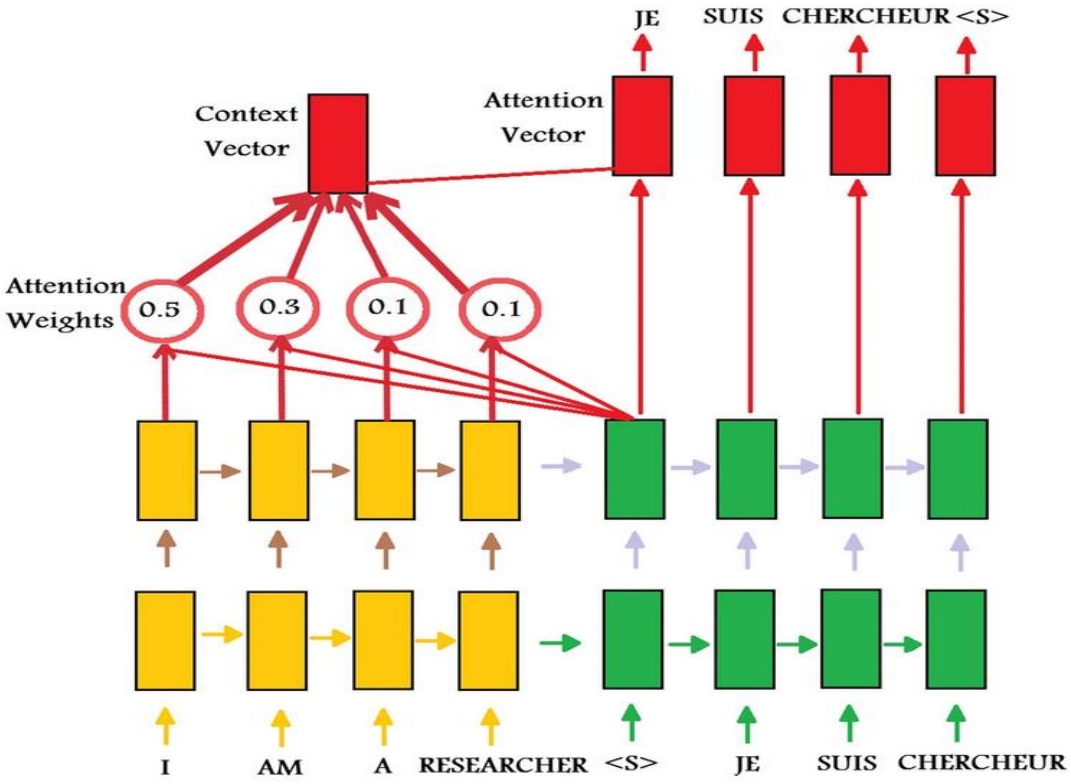

Figure 2.10 An attention mechanism's architecture (Laghrissi et al., 2021)

The attention mechanism was devised to empower the decoder to utilize the most pertinent elements of the input sequence. This provides flexibility by assigning higher weights to the most relevant encoded input vectors. Drawing inspiration from cognitive attention, This strategy has applicability in a variety of artificial intelligence models, encompassing domains such as natural language processing (Vasudevan, Dai, & Van Gool, 2021) and computer vision(Bian, Zhang, Zhao, Wang, & Gong, 2021). Initially

developed for Seq2Seq models in Neural Machine Translation, the fundamental seq2seq approach comprises an encoder-decoder model(Laghrissi et al., 2021). The encoder evaluates the input data and condenses the information into a fixed-length context vector (sentence embedding), while the decoder utilizes this context vector to generate the transformed output. While this architecture has demonstrated considerable efficacy in Seq2Seq tasks, it does suffer from a significant limitation. The sentence embedding is represented by a single vector, thereby posing challenges when dealing with longer input data. The model finds it more difficult to accurately capture all relevant.

2.4.2 Deep learning evaluation metrics

2.4.2.1 Confusion matrix-based performance measure

The majority of research in the intrusion detection field evaluates the efficacy of detection models using measures including F1-score, precision, recall, and accuracy. In a similar manner, we have used these metrics in our work. Table 2.2 presents the confusion matrix utilized in calculating these metrics.

Table 2.2 Confusion matrix for the intrusion detection system.

	Attack	**Normal Traffic**
Attack	Accurately classified attack samples. True Positives (TP)	Misclassified normal samples. False Negatives (FN)
Normal Traffic	Misclassified normal samples. False Positives (FP)	Accurately classified attack samples. True Negatives (TN)

- Accuracy represents the proportion of correctly classified normal and attack records relative to the total number of samples.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2.5}$$

- Precision specifies what percentage of successful predictions is accurate.

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

- Recall, also known as the Detection Rate, is the proportion of accurately classified attack samples.

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

- The F1 Score is a comprehensive metric that considers both Precision and Recall, providing an overall measure of how effectively the model detects attacks.

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

Finding the best model requires carefully choosing the right measure. When the cost of false positives is high, precision is important, whereas recall is preferable when the cost of false negatives is high. When seeking a balance between Precision and Recall, the F1 score serves as the correct metric to employ.

2.4.2.2 K-fold cross-validation

Typically, the dataset is divided into training and testing subsets in order to evaluate DL models. The testing set is used to assess the model's performance after it has been trained using the training set. The standard allocation involves dedicating 70 to 80% for training and 20 to 30% for testing, maintaining the association between labels and their respective features. Nevertheless, this approach lacks reliability as the evaluation metrics derived from one test set may vary from those obtained with a different test set.

The training dataset is split into k folds, each of equal size, in order to handle the problem with k-folds CV. The model is trained on one fold, and the other folds are

selected as the test set. In each iteration, a different fold serves as the test set to evaluate the model's error. Once the evaluation on one-fold is complete, the next fold in sequence is used, and the process is iterated to obtain another error estimate. This cycle continues until all k-folds have been employed. The final step involves computing the mean of the k error values. Reference Figure 2.11 provides a visual representation of the Cross-Validation process(Mueller & Massaron, 2021).

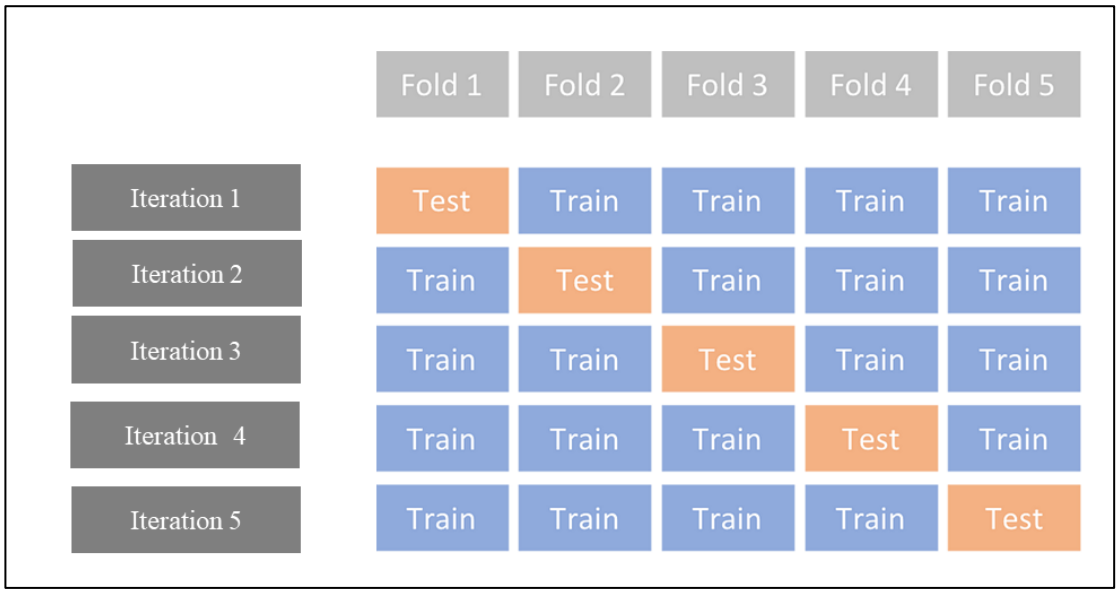

Figure 2.11 A visual interpretation of cross-validation

This chapter reviews SDN, Deep Learning, and NIDS. It discusses SDN's features, architecture, integration with NIDS, and security concerns. It also covers Deep Learning methods like CNNs, RNNs, and architectures such as AlexNet and LeNet5. The review highlights the potential of SDN and Deep Learning in enhancing network security but stresses the need to address associated challenges. It sets the stage for the development of a new approach in the next chapter.

3. METHODOLOGY

This chapter describes the proposed methodologies for NIDS in SDN. The preliminary models used in each of the methodologies are explained and the required descriptions are provided.

3.1 CNN-BILSTM: A Hybrid DL Approach for NIDS in SDN with Hybrid Feature Selection

In this work, we created a hybrid Convolutional Neural Network (CNN) and bidirectional long short-term memory (BiLSTM) model to enhance NIDS using multiclass classification. By balancing the dataset using the random over-sampling approach and selecting features using a random forest classifier along with the recursive feature elimination approach. The effectiveness of the proposed model was tested and assessed using the most frequently used datasets (UNSW-NB15 and NSL-KDD). In addition, we used the InSDN dataset, which is specifically dedicated to SDN.

3.1.1 The Proposed CNN-BiLSTM model

DL is an instance of ML. Using NNs with different hidden layers, the output and input layers of the models are constructed. Each input layer node receives metadata as input and processes the data through the hidden layers of the nonlinear transformation of the input data using training-adjusted weights to compute the output. One of the major purposes of DL is that it does not require the feature extraction phase. Precisely, instead of manually selecting and extracting features, DL learns the deep structure from the original data and extracts the features automatically.

However, the redundancy of noisy features and the huge amount of data in addition to the poor classification algorithm of deep learning cause the degradation of the accuracy of detection in the NIDS (Jie & Chengzhi, 2019). Therefore, in our proposed model, we used the RF classifier algorithm and RFE to select the optimal feature subset, as well as

deep learning to ensemble the output of the multi-classifier. As a result, the accuracy of Intrusion detection was improved effectively.

Figure 3.1 depicts the framework of the NID model in SDN, which includes these approaches. First, we start with transformation and cleaning that content normalization and numerical process. In addition, we balanced the dataset using random oversampling. Subsequently, we performed feature selection using a random forest classifier algorithm to obtain the highly important features. Finally, we deployed the hybrid model in the training and testing datasets to obtain multiclass classification results.

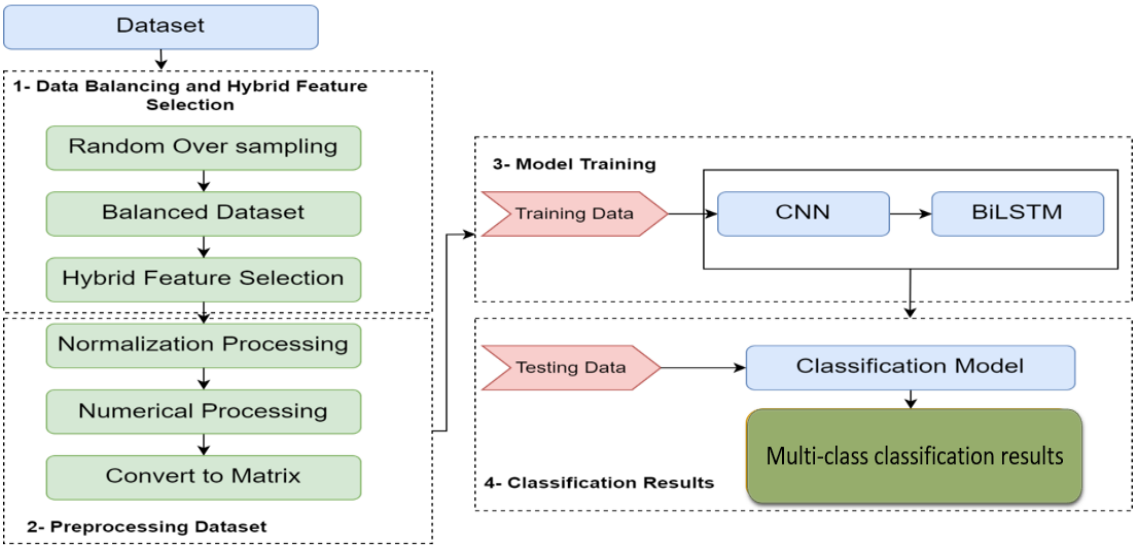

Figure 3.1 Architecture of proposed framework NIDS in SDN

This section describes the architecture of the CNN-BiLSTM hybrid model for learning and classifying network data in time and space. Figure 3.2 depicts the overall network architecture, which was divided into three phases. The first phase consists of feature selection, while the second and third phases are responsible for extracting and deriving data, respectively. Both CNN and BiLSTM are well-known deep learning algorithms. CNNs are capable of extracting spatial dimensional information from data. BiLSTM stands out due to its ability to preserve contextual background information over an extended duration, facilitating the extraction of data characteristics at the temporal level.

It is vital to examine the feature link at the spatial level while extracting features from a NIDS. As a result, this model integrates CNN and BiLSTM to extract features and then builds a hybrid model.

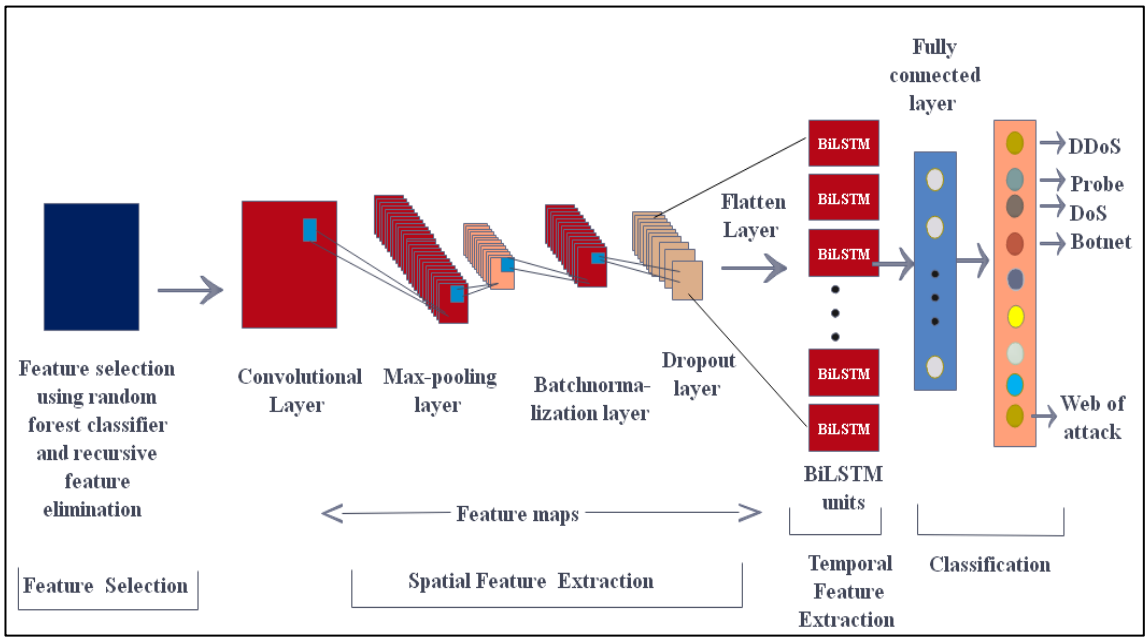

Figure 3.2 Architecture of the CNN-BiLSTM model

The generated spatial features are modified at the CNN output to comply with the BiLSTM network's input format as the CNN and BiLSTM mechanism inputs have distinct forms. In our instance, the output of the CNN's fully connected layer was 1 × 128, 1 × 64, and 1 × 32 feature vectors. When applied to the input layer of the BiLSTM model, the input value is fixed at 70. One layer of BiLSTM units was used to extract temporal features. Three convolution layers were created using kernel sizes of 5,3 and 5. A 2 × 2 Max Pooling layer is present in each convolution layer to reduce the dimensionality of the batch normalization and map features. The subsequent stage, which consists of three layers: a fully connected layer, a BiLSTM layer, and an output layer, these layers receive the high-dimensional features retrieved in the CNN stage. In both the BiLSTM and fully connected layers, there were 10 neurons for multiclass classification. Finally, the sigmoid layer was used at the output to indicate each input flow's probability for classification purposes.

We employed the L2 Regularizer and dropout approaches to mitigate the impact of overfitting and improve the detection model's capacity in unseen data. We ran multiple tests to find the appropriate regularization hyperparameter λ given the dropout technique's probability values. The value of λ is determined to be 0.1 for the L2 Regularizer, and a dropout with a probability P of 0.5 is employed after the following pooling layer and the entirely connected layer, respectively. We begin with a low dropout probability and progressively increase it because a dropout might induce some errors inside the learning model, and we want to limit the propagation of this loss to the subsequent layers. We trained all the models using k-fold cross-validation with 2-fold and 20 epochs for each fold.

- ***Hyper-parameters settings***

In some circumstances, the model may result in lower accuracy or even overfitting or underfitting. Performing hyperparameter adjustment is crucial for achieving high model performance. For that reason, the randomized search approach was utilized to refine the hyper-parameters and improve accuracy. Table 3.1 shows the hyperparameter values for the proposed model.

Table 3.1 Hyperparameters Settings

Parameters	Values
Kernels size	5.3.5
Filters	128.64.32
Size of the pool	2
Size of BiLSTM output	122
Regularization of the kernel	L2(0.1)
Weight restrictions	2
Optimizer	Adam
Activation function	Relu
Batch normalization	yes
Dropout	0.5
Loss function	Categorical_crossentropy for multi-class classification
Learning rate	0.001
Batch-size	128

3.1.2 Datasets for NIDS

In the realm of computer engineering, there exist several publicly available datasets containing records of both normal network traffic and instances of network attacks. These datasets are invaluable resources for scientists and professionals in the field. Each record within these datasets corresponds to a data packet in a network connection. These data packets are captured between defined starting and ending times, representing the movement of data among a source machine and a target machine, all transpiring under a specific network communication protocol.

To facilitate the analysis of network activities, these network connection data packets are stored as PCAP (Packet Capture) files, denoted by the pcapfile extension. It's noteworthy that PCAP files come in different formats tailored to various operating systems, such as Libpcap for Linux and macOS, WinPcap, and Npcap for Windows.

The significance of PCAP files lies in their utility for network analysis, traffic monitoring, and the management of security risks. They serve as a foundational element in identifying network-related issues, allowing for a comprehensive understanding of data usage patterns across applications and devices. Moreover, PCAP files play a crucial role in pinpointing security breaches, enabling the tracking of malicious traffic flow and other nefarious communications within the network. In essence, these datasets and the associated PCAP files serve as instrumental tools for enhancing both the proactive monitoring and reactive response to the dynamic landscape of network security challenges.

3.1.2.1 Description of the InSDN dataset

The InSDN Dataset is widely recognized as among the pioneering approaches for creating a comprehensive dataset aimed at examining IDSs for SDNs(Mahmoud Said Elsayed et al., 2020). The collection contains attacks with properties that are unique to SDN networks. Choosing an SDN-specific dataset is critical for conducting a thorough

evaluation of SDN-based threat detection techniques. Using Incompatible SDN datasets might lead to compatibility issues due to the necessity of aligning the attack vector implementation with the network's new architecture (Ring, Wunderlich, Scheuring, Landes, & Hotho, 2019). Furthermore, certain attacks operate differently on SDN networks than on traditional networks. For instance, an attacker may use techniques such as “IPsweep" and "Portscan" to flood the controller of SDN bandwidth using unidentified packets, which leads to a DDoS attack. However, the abovementioned attacks do not qualify as DDoS by traditional definitions, they may be utilized to produce a large number of data packets in an SDN network. Numerous records were included in the InSDN collection.

Figure 3.3 and Table 3.2 shows categorical attacks in the InSDN dataset, which includes both SDN-specific attacks on different types of normal traffic and attacks that are common to traditional networks. Among the various types of attacks are DoS, Web-attack, DDoS, Web, Botnet, Exploitation, Probe, and Password guessing or Brute force attacks. These are described as follows:

1. **DoS attacks**: Amongst the frequent attacks within the SDN architecture is the DoS attack. It might do so rapidly overload the SDN controller resource in addition to affecting the target system. Furthermore, the SDN controller functions as the brains of the SDN network; hence, a DoS attack renders the entire system inaccessible to authorized users. DoS attacks have the ability to inundate the target computer with a massive volume of spoof packets lacking matching criteria inside flow tables and switches. For additional processing, the OpenFlow switch will transmit these flows in the form of a packet-in message to the SDN controller. An excessive amount of untreated packets might overload SDN controller facilities when packet-in message rates are raised to a certain point(Mahmoud Said Elsayed et al., 2020).
2. **DDoS attacks**: A number of DDoS attack scenarios, including ICMP, UDP, and TCP-SYN flood attacks, are also included in the InSDN dataset. Among the most often utilized tools, Hping3 is utilized in DDoS attacks on the victim

computers, which are the Metasploitable 2 server and the h4 web server, and the attacker machines, h1 and h2(Mahmoud Said Elsayed et al., 2020).

3. **Password-guessing attacks or Brute force attacks**: Password-guessing attacks, also known as Brute force attacks, aim to gain access to the victim's computer by deciphering the password credentials and username. The InSDN dataset explores two distinct scenarios of Password-Guessing Attacks.
4. **Web application attacks:** A web attack targets vulnerabilities within websites with the aim of gaining unauthorized access, injecting malicious content, manipulating website content, or acquiring sensitive information. Websites present multiple potential attack surfaces, including web applications, CMS, web servers, and the underlying website code. Web applications enable users to interact with software via a web browser, such as filling out forms on a voter registration website. A CMS is a software application for managing, creating, and modifying website content, encompassing images, text, and layout. The web server consists of backend hardware and software that store website information and provide data to users. Two common web attacks are SQLi and XSS attacks.

- **SQL injection attacks** aim to insert custom SQL commands into website fields (e.g., password and username) to gain forbidden access to the data from the backend database.
- **XSS attacks:** Attackers using cross-site scripting attempt to insert and run illegal code inside of an online application. These attacks may cause unapproved text or images to be displayed, session credentials to be taken advantage of in order to impersonate users, or the attacker to get Forbidden entry to private data.

5. **Probe attacks:** Constitute a crucial preliminary stage for attackers before launching their assault. During this phase, the attacker systematically scans the target system to uncover valuable information that can aid in exploiting the remote system, including details such as operating system versions, open ports, and other relevant data.
6. **Botnet attack**: Even if a lot of equipment or gadgets connect to the Internet, the security that is offered does not ensure that it will work as best it can to eliminate intrusion attempts. The attacker has the ability to take over several compromised devices, often known as a botnet, and use them to carry out

various malicious operations including web application servers, cybercrime, fraud attacks, or DDoS attacks on target servers(M. S. Elsayed, N. A. Le-Khac, & A. D. Jurcut, 2020).

7. **U2R (Exploitation) attack**: Backdoor and remote exploitation attacks are regarded to correspond with the U2R example in the InSDN dataset. It is crucial to identify these attacks as soon as feasible since the malicious activity might pose a major risk to the network system and is more like to regular traffic (Sharma & Mukherjee, 2012).
8. **Normal traffic**: The authors(Mahmoud Said Elsayed et al., 2020) take into account actual Internet traffic that uses many protocols, including mail, DNS, SSH, HTTP, HTTPS, and so on. The h3 host is linked to the Internet and runs several apps, such as email, twitter, facebook, and Skype voice, in order to produce more inherent traffic. In addition, a number of services, including SSH, FTP, Telnet, and others, are visited on the Metsploitable2 server in order to create different samples for typical traffic.

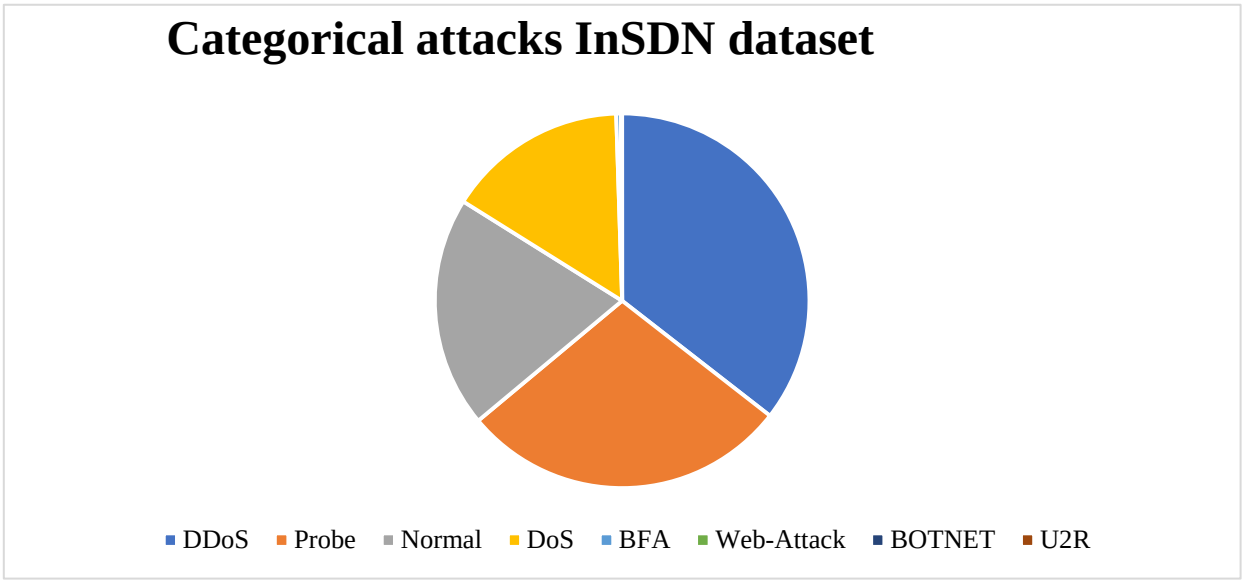

Figure 3.3 Categorical attacks InSDN dataset

Table 3.2 Frequency distribution of attack classes in SDN dataset: Training vs. Testing

Category	Attack class	Frequency train (%)	Attack class	Frequency test (%)
DDoS	81843	35.52	40099	35.33
Probe	65488	28.42	32641	28.76
Normal	45957	19.95	224667	19.80
DoS	35943	15.60	17673	15.57
BFA	927	0.40	478	0.42
Web-Attack	127	0.06	57	0.05
BOTNET	107	0.05	57	0.05
U2R	13	0.01	4	0.01

3.1.2.2 Description of UNSW-NB15 dataset

UNSW-NB15 dataset, which includes the most current revisions, was created by the research team at the Australian Centre for Cyber Security(Moustafa & Slay, 2015) to address the concerns detected in the NSL-KDD and KDDCup 99 datasets. Figure 3.4 and Table 3.3 shows the categorical attacks in UNSW-NB15 dataset, the dataset consists of 82337 test set and 175343 train set with 9 attacks. It has included 42 characteristics with parallel-class labels that were typical and nine distinct intrusions. These are as follows:

- **Analysis**: Various port scan, spam, and HTML file penetration attacks(Shah, Khan, & Ashraf, 2020).
- **Backdoors**: System security mechanisms are bypassed to stealthily access a computer or its data(Moustafa & Slay, 2015).
- **Exploits**: Targeting a known security problem within an operating system or piece of software(Shah et al., 2020).
- **Reconnaissance:** Contains strikes that can simulate attacks that can be used to gather information(Shah et al., 2020).

- **DDoS:** An attacker employs a distributed form of attack by inundating a server with an extensive volume of malicious requests. The legitimate traffic overwhelms the computing and memory resources, leading to a state of busyness that obstructs access for genuine users(Dhanabal & Shantharajah, 2015).
- **Fuzzers**: sending randomly generated data in an effort to interrupt a program or the network(Shah et al., 2020).
- **Worms:** A self-replicating code or program that leverages security breaches to propagate over a network. (Shah et al., 2020).
- **Shellcode:** A small piece of code used as a payload to exploit a vulnerability(Shah et al., 2020).
- **DoS:** an effort to stop a host's services permanently in order to prevent users from accessing a server or resource(Shah et al., 2020).

- **Generic**: A technique that works against block ciphers without consideration of the structure of the block cipher (Shah et al., 2020).

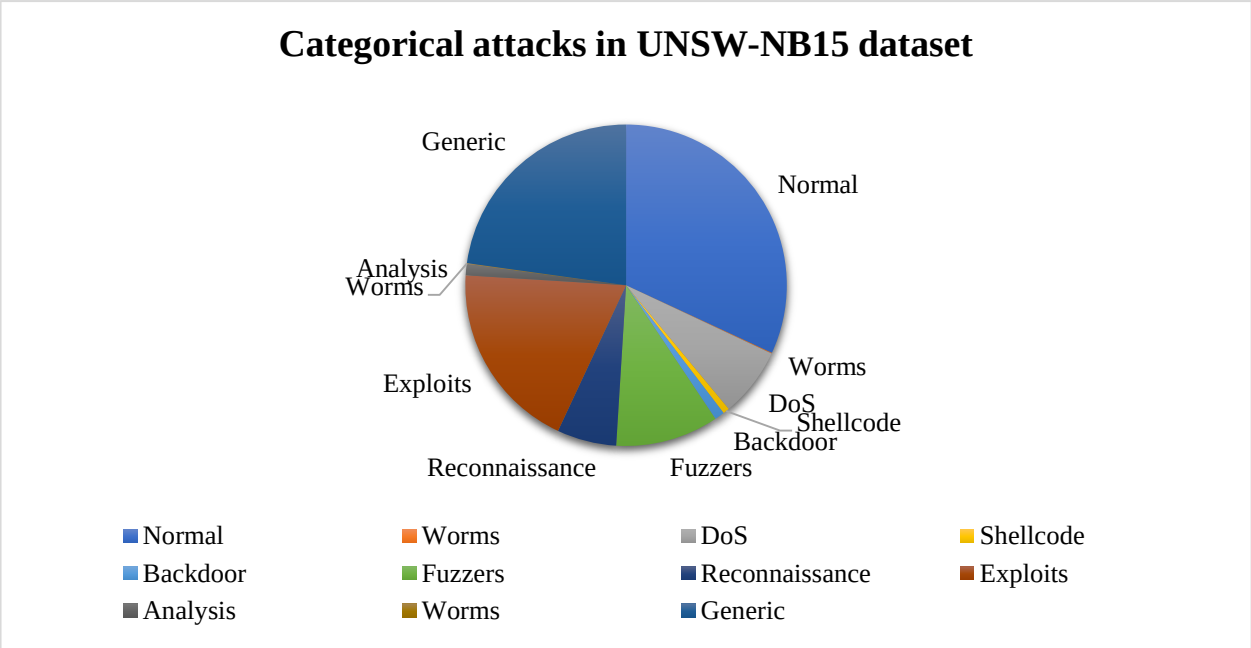

Figure 3.4 Categorical attacks in UNSW-NB15

Table 3.3 Comparison of attack categories in train and test of UNSW-NB15 dataset

Category	Train	Test
Dos	11656	9711
Probe	45927	2421
R2L	995	2754
U2R	52	200
Normal	67343	9711
Total	125973	22544

3.1.2.3 Description of the NSL-KDD dataset

Table 3.4 Comparison of attack categories in train and test of NSL-KDD dataset

Category	Train	Test
Normal	56000	37005
DoS	12264	4089
Shellcode	1133	378
Backdoor	1746	583
Fuzzers	18185	6062
Reconnaissance	10492	3496
Exploits	33393	11132
Analysis	2000	677
Worms	130	44
Generic	40000	18871
Total	125973	22544

The NSL-KDD and the KDD CUP99 are well-known datasets in the domain of NID, and Revathi's research demonstrates that the NSL-KDD datasets are ideal for evaluating different intrusion detection algorithms(Revathi & Malathi, 2013). Figure 3.5 and Table 3.4 shows categorical attacks in the NSL-KDD dataset., Each incursion record in this

dataset includes a 42-dimensional feature that is subdivided into a traffic-type label, a 3D symbol feature, and a 38D digital feature. The label mostly comprises normal data as well as the data of four categories of attack (U2R, DoS, Probe, and R2L).

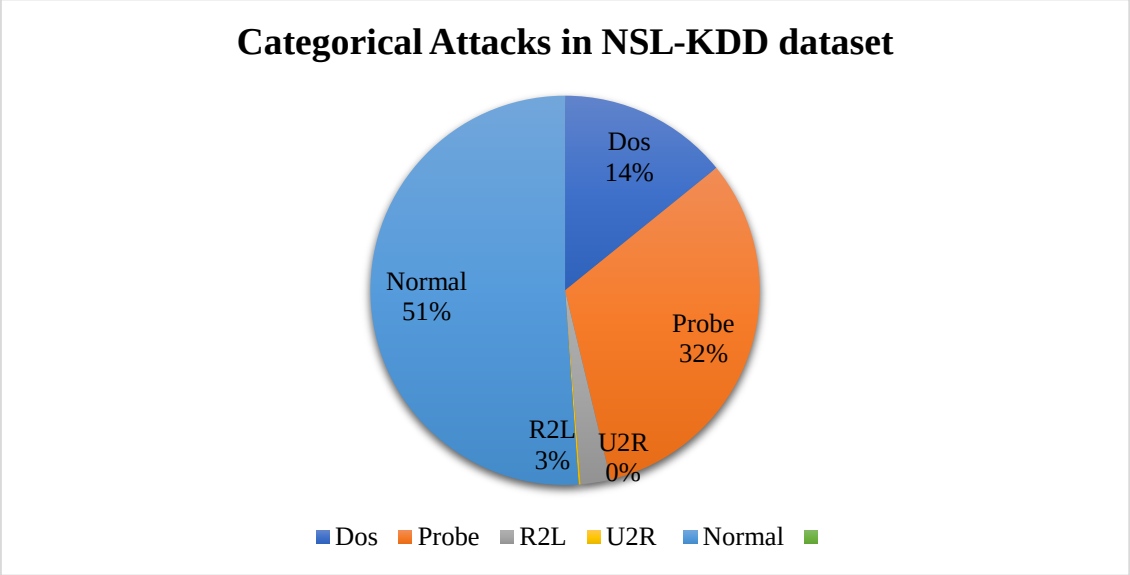

Figure 3.5 The NSL-KDD dataset's attack types

The NSL-KDD dataset's attacks are divided into four groups:

- **R2L**: Unauthorized access to a local area network from a distant computer.
- **DoS**: Denial of Service, caused by an excessive volume of traffic overloading the system.
- **Probing**: Using surveillance and additional probing techniques to get data from a network.
- **U2R**: An unauthorized user using a regular user account as a super-user (gain access to the root)

The subclasses for each attack are shown in Figure 3.6, resulting in 39 attacks:

Classes:	DoS	Probe	U2R	R2L
Sub-Classes:	• apache2 • back • land • neptune • mailbomb • pod • processtable • smurf • teardrop • udpstorm • worm	• ipsweep • mscan • nmap • portsweep • saint • satan	• buffer_overflow • loadmodule • perl • ps • rootkit • sqlattack • xterm	• ftp_write • guess_passwd • httptunnel • imap • multihop • named • phf • sendmail • Snmpgetattack • spy • snmpguess • warezclient • warezmaster • xlock • xsnoop
Total:	11	6	7	15

Figure 3.6 Subclass of each attack in the NSL-KDD dataset

3.1.3 Dataset preprocessing

The process of data processing, which is referred to as data engineering, is crucial for the success of the learning process. It involves various procedures such as column and row cleaning, feature encoding, and data normalization, which are used to preprocess the data to ensure that they are adequately prepared for analysis. All these procedures were applied to all datasets. This subsection discusses the details of these procedures as follows.

3.1.3.1 Drop rows and missing values

To ensure data integrity, all rows were inspected thoroughly to detect missing values. This process is a standard practice in data preparation, as the "id" column typically contains no significant data. Eliminating the "id" column allows for a focus on examining only essential features.

3.1.3.2 Catgeorical labelEncoder cnconding

The process of categorical transformation holds significance in enhancing the learning capacity of classifiers that are designed to handle only numerical data. In the case of the InSDN dataset, for example, characteristics such as "proto," "service," and "state" refer to numerical representations of category data.

We have opted to employ the LabelEncoder technique for encoding purposes(Bisong, 2019). This option gives each unique category value a unique number label, making it ideal for the InSDN dataset. Through this transition, the classifier is able to understand and learn from the encoded labels. While LabelEncoder differs from OneHotEncoder in that it does not generate distinct binary columns, it nonetheless preserves the categorical characteristics of the feature(Jackson & Agrawal, 2019).

3.1.3.3 Balancing the dataset

The easiest oversampling technique for balancing an unbalanced dataset is random oversampling when compared to other techniques. The minority class samples are duplicated to establish data parity. This does not result in any information loss; however, the dataset is susceptible to overfitting because identical information is duplicated. In this study, we used random oversampling to address dataset imbalances.

3.1.3.4 Hybrid feature selection

Data were gathered from network packets to identify intrusions. As a result, manually classifying the huge quantity of network data gathered by the system is a time-consuming operation. Apart from gathering network data, evaluating it is a complicated process due to the number of behavioral patterns and attributes included in the data. To protect the network from attacks, real-time intrusion detection is necessary, which may be accomplished by mining accessible datasets for key characteristics. The reduced collection of features can significantly improve the intrusion detection rate (J. Li et al.,

2017). The selection of features may be accomplished in a variety of ways. For example, filtering data that are unnecessary to the detection process in order to categorize the attacks, clustering data based on their similarity in order to uncover hidden patterns in the data for classification, and deleting irrelevant features from the feature set using feature selection techniques.

To choose the most significant features, we employed the RF classifier technique, which is very accurate and effective with large datasets.(Jie & Chengzhi, 2019). RF is a classifier composed of several decision trees. It is a mixture of classification trees so that each tree is reliant on the values of a random vector selected randomly for all trees in the forest and having the same distribution(Breiman, 2001). When new records are received as input, the RF stores them in forest trees. Each tree provides a categorization, and the forest selects the class in which the majority of trees fall(Breiman, 2001). From the results of the RF classifier, we selected 10 features using RFE because of the high accuracy achieved after different attempts while training our proposed model using different sets of selected features. Figure 3.7 shows the importance of each feature and Table 3.4 shows the feature name and the descriptions of the 10 selected features.

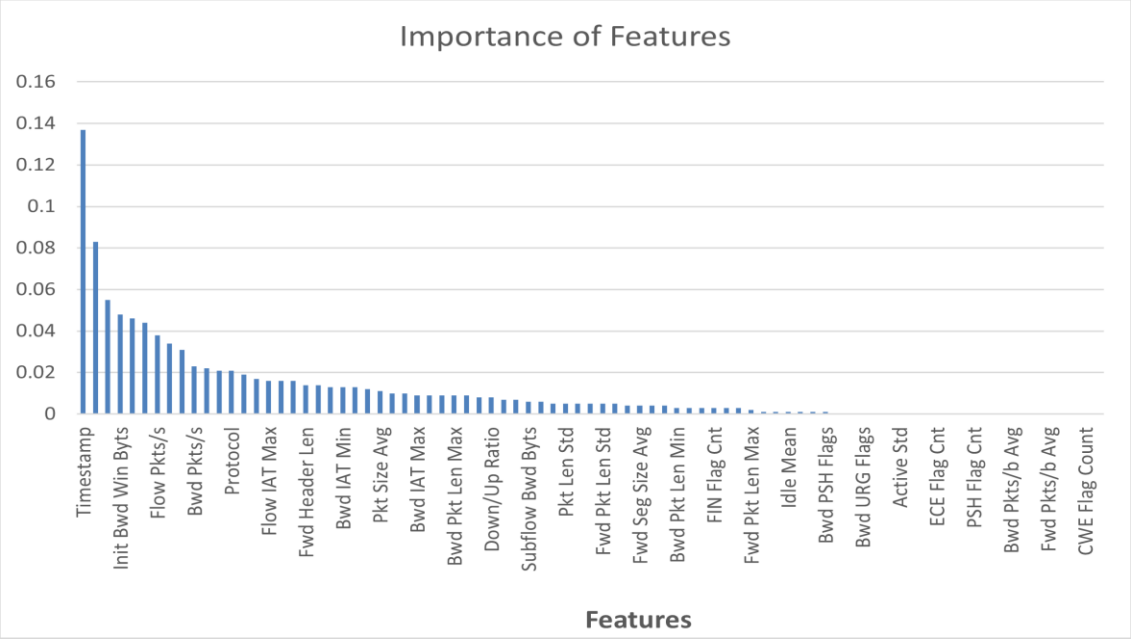

Figure 3.7 High importance of the features

Table 3.5 Selected 10 features with their description

Feature name	Description
'Timestamp',	Records the date and time when a particular event or data entry occurred.
'Init Bwd Win Byts',	The total number of bytes sent in the initial window in the backward direction.
'Flow Pkts/s',	Number of flow Packets per second.
‘Bwd Pkts/s,	Number of backward packets per second.
'Dst Port',	Destination port number
‘Protocol’,	The protocol type
'Src Port',	Source port number
'Dst IP',	Destination IP address
'Fwd IAT Min',	Min and standard deviation of the time between two packets sent in the forward direction.
'Bwd Header Len',	Total bytes used for headers in the backward direction.

3.1.3.5 Normalization processing

The calculated value scope of continuous feature data in the InSDN dataset is noticeably different; in the dataset, the value scope of logged-in is [0,39], whereas the value scope of num compromised is [0,27]. As a result, the maximum value scope varies greatly. To make arithmetic calculations and dimension removal easier, the value scope of any feature inside the [0,1] interval is consistently and mapped using the normalization processing technique. The normalization equation's formula (1) is as follows:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

Where X_{norm} represents feature nnormalized X_{max} is the maximal feature's value and X_{min} is the minimal value.

3.1.3.6 Numerical processing

One-hot encoding was used since the model's input is a digital matrix. The benefit of this approach is that it creates a digital feature vector by matching symbol characteristics in the dataset's data.

3.1.3.7 Generating a matrix using normalized data.

Each network is recorded and converted to a grayscale image format. Subsequently, the network data is reformed into a matrix to enter the CNN. For instance, it mentions that 100 feature vectors are reformed into a 10×10 matrix.

3.1.4 Experimental setup

In the context of dataset analysis and model design, a set of powerful tools is employed, leveraging both hardware and software components. The computational infrastructure comprises different elements used in The Truba platform from TUBITAK ULAKBIM is shown in Table 3.6 below:

Table 3.6 Experimental setup for model training.

Project	Environment
GPU	12 core x 2 CPU + 2x Nvidia M2090
Memory	256 GB + 6 GB GDDR5
CPU	Xeon E5-2680 v3 2.50 GHz
Description	Levrek-CUDA Cluster
Framework	Keras2.2
Architecture:	*x86_64*
CPU op-mode(s):	*32-bit, 64-bit*
CPU(s):	*48*
Vendor ID:	*GenuineIntel*
CPU family:	*6*
Model name:	*Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz*

The toolkit leverages a suite of powerful tools and libraries to facilitate seamless data analysis, manipulation, and visualization including the following:

- The Jupyter Notebook, an open-source web application and programming environment, serves as a pivotal tool in this process. Compatible with various programming languages, including Python, it facilitates a collaborative and interactive approach to data analysis. Python, being a free general-purpose programming language with an elegant syntax, is chosen as the primary programming language for its versatility and platform compatibility across Linux, Windows, and Mac OS. Python's prominence in modern technologies, such as ML, robotics, AI, and data science, underscores its relevance and efficiency, surpassing many older languages in these domains (Shovic & Simpson, 2019).

- To streamline data manipulation and analysis, the toolkit includes Pandas, a Python library offering simple, versatile, and expressive data structures. Pandas serves as a foundational component for practical, real-world data analysis, striving to be the most efficient open-source tool across various programming languages. NumPy, another essential library, supports scientific computing in Python, providing efficient broadcasting tools for seamless integration with C/C++ and Fortran code.

- For visualization purposes, the toolkit employs Matplotlib, an extensive Python library capable of constructing static, animated, and interactive visualizations. In the realm of ML, Scikit-learn emerges as a popular choice, offering a comprehensive suite of algorithms for tasks such as classification, clustering, regression, and dimensionality reduction. Beyond algorithms, it encompasses modules for data preprocessing, feature extraction, hyperparameter optimization, and model evaluation.

- TensorFlow, is an open-source machine learning platform that serves as a cornerstone in the toolkit. It offers a robust and scalable ecosystem of

libraries, tools, and community resources. TensorFlow empowers researchers to advance the frontiers of state-of-the-art ML techniques, while developers benefit from its efficiency in creating and deploying machine learning-powered applications. In conjunction with TensorFlow, Keras, a Python-based deep learning library, operates atop TensorFlow's framework. Keras emphasizes rapid experimentation and facilitates a seamless transition from concept to fruition in the research and development pipeline. Together, TensorFlow and Keras provide a powerful combination, enabling both researchers and developers to achieve their machine-learning goals effectively.

3.2 Attention-based CNN-BILSTM DL Approach for NIDS in SDN

In this study, we propose the intrusion detection hybrid model based on CNN and a BiLSTM with an attention mechanism. The model consists of three main components: a CNN layer, a BiLSTM layer, and an attention layer. The CNN layer extracts local features from the network traffic data. The BiLSTM layer learns the temporal dependencies between the local features. The attention layer selects the most relevant features from the BiLSTM output for each intrusion type. Our hybrid model can effectively detect a wide range of intrusions, including Brute force, Web attacks, and DDoS. The hybrid model has several advantages over the state-of-the-art intrusion detection models. First, our model can effectively capture complex network traffic patterns. Second, it can identify intrusions with high accuracy. Third, it is efficient and can be easily deployed in SDNs. We evaluate our model on a real-world SDN dataset (InSDN dataset) and an NSL-KDD dataset. The experimental results show that our hybrid model outperforms the state-of-the-art intrusion detection models in terms of accuracy, precision, recall, and F1 score like Alexnet, Lenet5, CNN, and CNN-LSTM models.

3.2.1 The proposed attention-based CNN BiLSTM model

The Attention-based CNN BiLSTM model combines the strengths of CNNs in capturing local patterns and spatial information, Bi-LSTMs in capturing long-range dependencies, and attention mechanisms in focusing on important parts of the sequence. This architecture has been proven to be effective in various tasks that involve sequential data, offering improved performance and interpretability compared to traditional DL models.

The suggested model's architecture is depicted in Figure 3.8. The input is subsequently processed by a CNN layer comprising 128 filters and a kernel size of 5, employing the rectified linear unit (ReLU) activation function for introducing non-linearities. This layer serves to capture local patterns and extract relevant features from the input data. Following the CNN layer, a MaxPooling operation is performed to downsample the feature maps and reduce spatial dimensions.

Concurrently, the input is fed into a BiLSTM layer, consisting of 128 units. By utilizing a bidirectional configuration, the BiLSTM layer is capable of modeling temporal dependencies and capturing sequential patterns in the input sequences. This enhances the model's ability to understand the context and dynamics of the data.

To incorporate the attention mechanism, the BiLSTM outputs are further processed using a TimeDistributed Dense layer. This layer applies non-linear transformations to each time step's output, enabling it to learn the relevance and significance of different time steps. The transformed outputs are then subjected to a Tanh activation function, generating attention weights that indicate the relative importance of each time step. These weights are expanded and permuted to align with the subsequent steps of the attention mechanism.

The attention weights are then used to modulate the BiLSTM outputs via element-wise multiplication. This operation produces a weighted representation that highlights the

salient features within the input sequence. The weighted outputs are subsequently aggregated through a summation operation along the time axis, yielding a fixed-size representation that encapsulates the most pertinent information from the input.

The flattened output from the CNN layer and the attention-weighted BiLSTM representation are concatenated, allowing the model to effectively leverage both local and global features for making predictions. To mitigate overfitting, a Dropout layer with a dropout rate of 0.5 is introduced, randomly deactivating connections during training. Finally, a Dense layer employing the softmax activation function is applied to produce class probabilities for the multi-class classification task. The model is trained to optimize the specified loss function using the Adam optimizer, with the primary objective of achieving high accuracy in predicting the correct class labels for the input sequences.

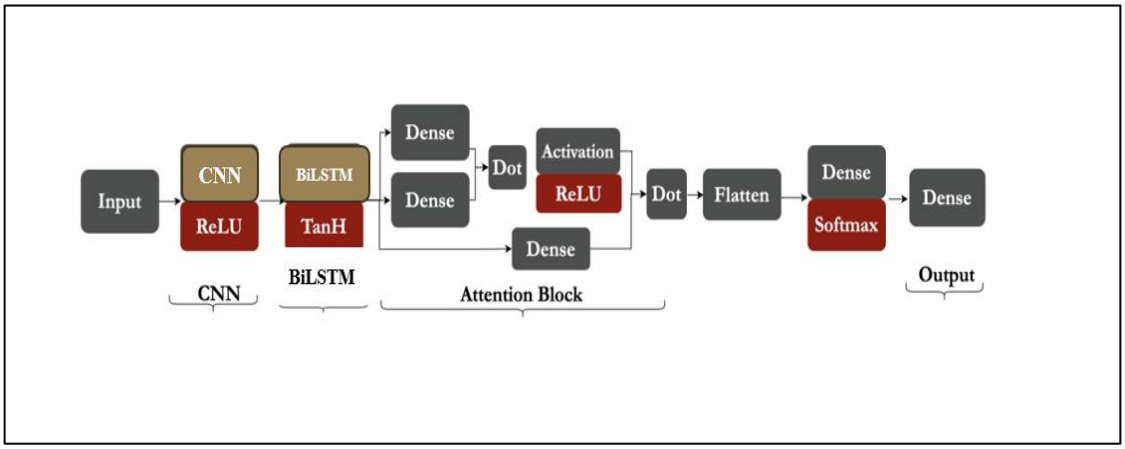

Figure 3.8 Attention-based CNN-BiLSTM model architecture

- **Hyper-parameters setting**

In some circumstances, the model may result in lower accuracy or even overfitting or underfitting. Performing hyperparameter adjustment is crucial for achieving high model performance. For that reason, the randomized search approach was utilized to refine the hyper-parameters and improve accuracy. Table 3.7 shows the hyperparameter values for the proposed model.

Table 3.7 Hyperparameters Setting of the hybrid model

Parameters	**Values**
Kernels size	5,3,5
Filters	128,64,32
Size of the pool	2
Size of BiLSTM output	122
Regularization of the kernel	L2(0.1)
Weight restrictions	2
Optimizer	Adam
Activation function	softmax
Batch normalization	yes
Dropout	0.5
Loss function	Categorical_crossentropy for multi-class classification
Learning rate	0.001
Batch-size	128

3.2.2 Experimental setup

We used Google Colab to run our experiments. A free cloud-based Jupyter Notebook environment called Google Colab enables you to execute Python code without having to install any software on your machine. There was a Tesla P100-PCEI-16GB GPU, an Intel(R) Xeon(R) CPU @ 2.30GH processor, and 12 GB RAM. This makes it a convenient and easy-to-use platform for running experiments, especially for those who do not have access to a powerful computer.

In our experiments, We employed the InSDN, and NSL-KDD datasets to test the hybrid model. We split the datasets to 70% for training and 30% for testing. All models are run with K-fold CV with 5-fold and 10 epochs for each fold. To train our hybrid model, we set the dropout to 0.5, the batch size was 128, and the optimizer used is Adam, for the loss function was categorical cross entropy and the activation function was softmax for multi class classification.

3.2.3 Data preprocessing

A crucial step in learning-based research is data preprocessing, which guarantees the reliability and correctness of the analysis. In this study, we suggest a methodology for data preprocessing that consists of the following steps:

In order to prepare the data for utilization by the classification models, data preprocessing is carried out. Intrusion detection datasets contain network features in diverse data formats, necessitating the conversion of these network features into a consistent format for subsequent processing. Feature encoding and feature normalization techniques are employed to preprocess the network features. Categorical features undergo a transformation into numerical features by employing the one-hot encoding technique. This ensures that the data is appropriately formatted and ready for further steps which is deploying the hybrid model to get classification results.

4. RESULTS AND DISCUSSION

This chapter presents the experimental results and a performance of two hybrid deep learning approaches for NIDS in SDN with other DL models like CNN, CNN-LSTM, AlexNet, and LeNet5. The assessments make use of the three datasets that were discussed in the preceding chapter. Performance metrics that are used for assessment include f1-score, recall, accuracy, and precision.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{4.2}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{4.3}$$

$$\text{F1} - \text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

4.1 CNN-BILSTM: A Hybrid DL Approach for NIDS in SDN with Hybrid Feature Selection

4.1.1 InSDN dataset results

Table 4.1 provides detailed findings for multi-class classification using several classifiers on the InSDN dataset. In terms of different metrics, the performance of various attacks varies significantly amongst classifiers.

- The accuracy of several models was illustrated in Figure 4.1, with the Proposed CNN-BiLSTM model providing the best accuracy from among the models examined. It outperforms the other models with an accuracy of 97.12%. The CNN-LSTM model

follows closely with an accuracy of 96.69%. LeNet5 and AlexNet models achieved accuracies of 93.97% and 89.79% respectively, while the CNN model had the lowest accuracy at 85.94%.

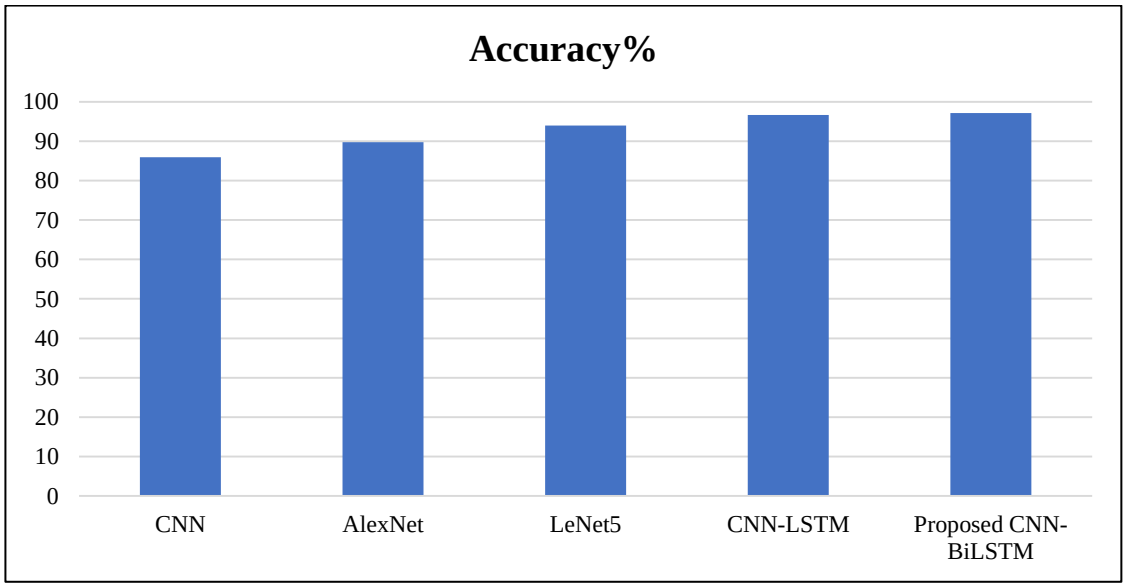

Figure 4.1 Accuracy results of multi-class classification on InSDN dataset for different models

- The models exhibited a range of recall values, spanning from 45% to 100%. Among them, the proposed CNN-BiLSTM model achieved the highest recall rate of 100% for the U2R and R2L attack classes. Additionally, the CNN-LSTM model attained the highest recall rate of 100% for the U2R class and 100% for R2L.
- In terms of precision, the models demonstrated values ranging from 6.36% to 100%. Notably, the proposed CNN-BiLSTM model achieved the highest precision score of 100% for the Web Attack and Probe classes, while the CNN-LSTM model attained the highest precision score of 100% for the U2R and R2L classes. Finally, the LeNet5 model achieves 100% for the Probe class.

- Figure 4.2 shows the F1-score of all models. The analysis of F1-scores across diverse attack classes reveals significant performance disparities among the models. Specifically, for the U2R class, the CNN-LSTM model achieves the highest F1-score (100%), followed by the proposed CNN-BilSTM model

(99.22%) and LeNet5 (26.79%). Similarly, in the R2L class, the CNN-LSTM model leads with the highest F1-score (100%), closely followed by the proposed model (99.98%) and LeNet5 (91.98%), showcasing its robustness in capturing instances of R2L attack. For the other models like AlexNet, CNN achieved 26.79 %, and 37.16 % respectively.

Regarding the Probe class, while the proposed CNN-BiLSTM model achieves the highest F1-score (100%), the LeNet5 model secures the second-highest score (99.98%), underlining its proficiency in identifying probing activities. Although all models exhibit exceedingly high F1-scores (> 60%) for the DoS class, the proposed CNN-BiLSTM model secures the first-highest score (96.18%), reaffirming its capability in detecting DoS. Similarly, for the Normal class, where all models achieve very high F1-scores (> 90%), the AlexNet model secures the first-highest score (97.61%), highlighting its reliability in discerning normal network traffic. Overall, the Proposed attention-based CNN-BiLSTM model demonstrates commendable performance across all classes, particularly excelling in the Probe class.

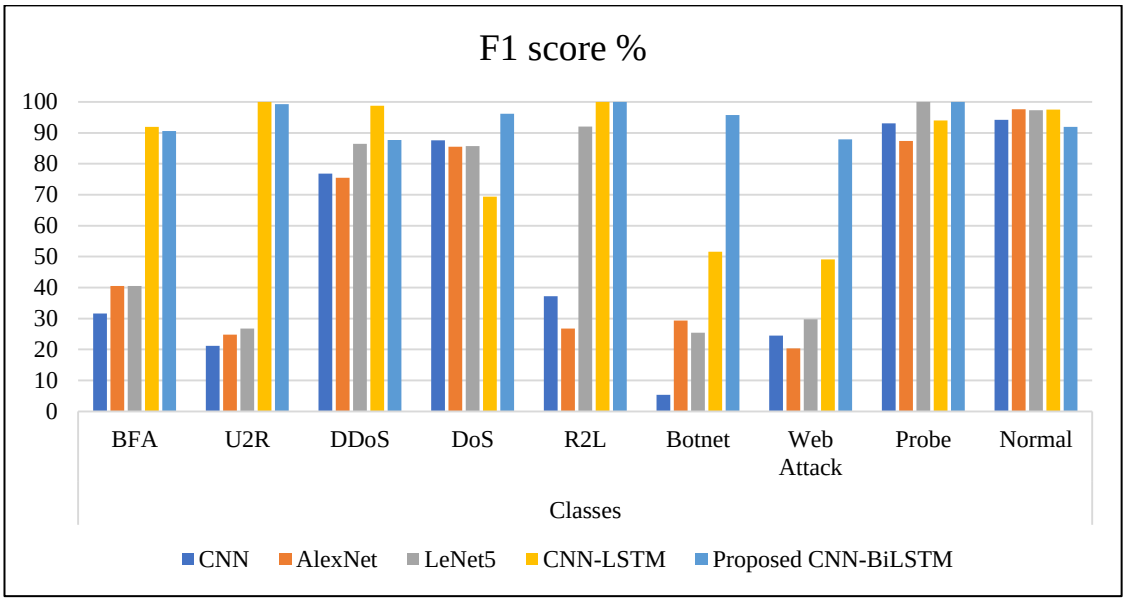

Figure 4.2 All models of multiclass classification on the InSDN dataset were scored using the F1 score metric.

For the majority of the classes, the suggested CNN-BiLSTM model had the best f1-score, accuracy, recall, and precision overall. This indicates that the best model for classifying network attacks is the proposed CNN-BiLSTM model.

Table 4.1 Results of InSDN dataset multi-class classification

Models	**Metrics (%)**	**Class**								
		BFA	**U2R**	**DDoS**	**DoS**	**R2L**	**Botnet**	**Web Attack**	**Probe**	**Normal**
CNN	Accuracy	**85.94**								
	Recall	54.74	60.84	85.41	90.25	50	7	16.25	94.86	98.10
	Precision	15.87	50.91	74.15	85.45	18.65	6.36	50	94.69	96.33
	F1-Score	31.65	21.12	76.86	87.61	37.16	5.36	24.52	93.05	94.20
AlexNet	Accuracy	**89.79**								
	Recall	50.54	30.56	80.42	84.25	49.92	48.25	46.25	87.90	97.85
	Precision	48.35	23.85	73.99	88.98	13.43	14.72	10.21	89.85	95.44
	F1-Score	40.54	24.79	75.45	85.47	26.79	29.36	20.37	87.38	97.61
LeNet5	Accuracy	**93.97**								
	Recall	50	50.84	80.64	80.47	92.89	20.38	26.69	99.99	99.71
	Precision	20.35	13.43	83.75	82.25	94.97	26.96	28.36	100	96.58
	F1-Score	40.54	26.79	86.39	85.68	91.98	25.45	29.78	99.98	97.26
CNN-LSTM	Accuracy	**96.69**								
	Recall	90.18	100	99.38	65.12	100	56.73	51.04	98.77	98.45
	Precision	92.42	100	99.78	66	100	51.66	50.88	95.4	97.21
	F1score	91.89	100	98.74	69.37	100	51.53	49.12	93.97	97.51
Proposed CNN-BiLSTM	Accuracy	**97.12**								
	Recall	95.99	100	90	93.75	100	94.87	78.4	99.87	90.18
	Precision	93.43	98.46	85.61	99.03	99.83	96.36	100	100	92.42
	F1-Score	90.58	99.22	87.7	96.18	99.98	95.73	87.89	100	91.89

4.1.2 UNSW-NB15 dataset results

The performance of several models on an attack detection task is shown in Table 3.9. The models are evaluated using different measures, including f1-score, recall, precision, and accuracy. The suggested CNN-BiLSTM model has the highest accuracy of 84.23%. It also has the highest F1-score and recall for most of the classes. This suggests that the proposed model can effectively detect attacks with high accuracy.

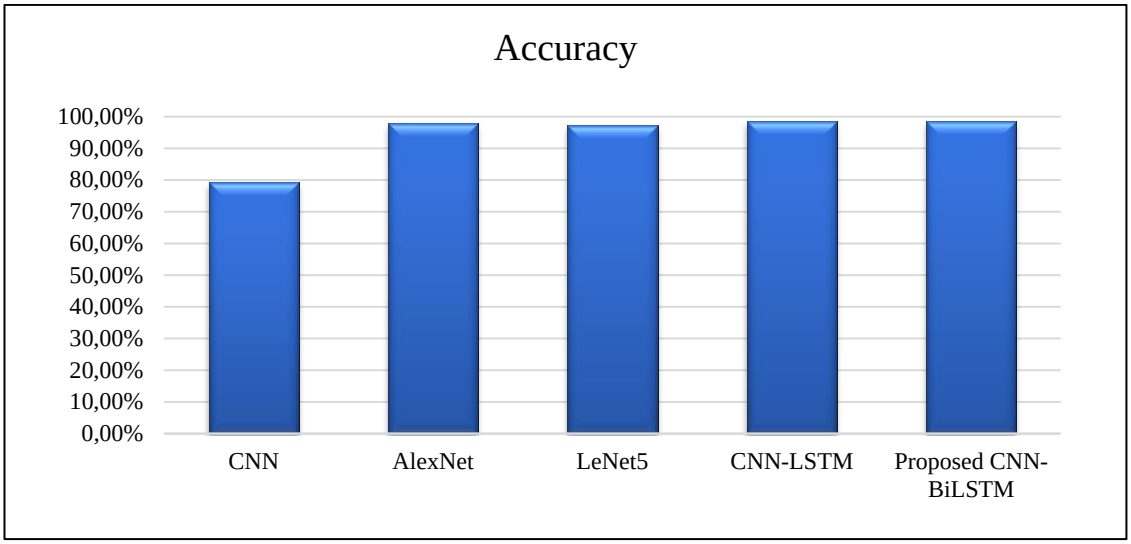

Figure 4.3 Results of multiclass classification accuracy for various models on the NSL-KDD dataset.

Table 4.2 Outcomes of the multiclass classification using UNSW-NB15

Models	**Metrics (%)**	**Class**								
		Backdoor	**Analysis**	**Fuzzers**	**Shellcode**	**Reconnaissance**	**Exploits**	**DoS**	**Worms**	**Normal**
CNN	Accuracy	**67.41**								
	Recall	33.45	10.36	50.01	78.65	96.73	92.26	73.08	73.76	45.89
	Precision	26.69	15.6	43.83	65.45	80.77	78.14	48.64	98.32	36.15
	F1-Score	34.8	13.49	80.55	83.24	88.03	73.13	58.41	84.29	53.10
AlexNet	Accuracy	**77.54**								
	Recall	82.36	80.25	66.89	80.76	63.37	88.87	81.64	92.99	87.05
	Precision	76.54	85.73	54.02	67.01	60.79	61.41	77.15	91.34	81.68
	F1-Score	81.39	77.48	51.32	70.7	62.72	65.92	88.69	88.76	88.70
LeNet5	Accuracy	**76.35**								
	Recall	76.59	74.95	47.89	87.48	59.92	70.69	77.99	72.68	87.29
	Precision	74.95	52.21	49.09	66.49	57.67	58.15	71.94	73.46	91.86
	F1-Score	79.45	71.19	15.39	69.86	60.98	59.86	74.25	72.98	89.52
CNN-LSTM	Accuracy	**79.92**								
	Recall	65.25	76.12	55.59	26.06	69.81	88.47	55.00	78.72	91.88
	Precision	63.89	60.05	58.22	61.19	72.98	59.91	51.92	88.50	89.41
	F1-Score	67.58	61.15	56.87	36.55	71.36	71.44	10.08	59.10	90.63
Proposed CNN-BiLSTM	Accuracy	**84.23**								
	Recall	49.29	54.95	68.69	86.08	97.98	93.4	79.31	90.72	91.47
	Precision	49	42.56	54.29	66.83	91.2	75.45	72.31	88.87	92.35
	F1-Score	44.35	42.18	51.16	70.51	94.01	81.82	74.88	88.81	90.58

Table 4.2 is a more detailed breakdown of the results for the UNSW-NB dataset for the proposed model:

- Figure 4.3 shows the accuracy of different models, In the context of overall accuracy, the Proposed CNN-BiLSTM model emerges as the top performer, boasting the highest accuracy rate of 84.23%. Following closely, the CNN-LSTM model secures the second-highest accuracy at 79.92%, while AlexNet trails slightly behind with an accuracy of 77.54%. The LeNet5 model achieves the fourth-highest accuracy, standing at 76.35%, and finally, the CNN model exhibits the lowest overall accuracy among the evaluated models, with a rate of 67.41%.
- Recall: The performance evaluation of various models unveils nuanced insights into their effectiveness across different attack classes. Despite achieving the highest overall accuracy (84.23%), the proposed CNN-BiLSTM model exhibits lower recall rates for certain classes compared to its counterparts. In contrast, AlexNet demonstrates the most consistent performance with the best average recall (82.36%) across all classes, closely followed by LeNet5 (76.59%). Specifically, CNN encounters challenges in accurately classifying some classes, particularly DoS (73.08%) and Normal (45.89%). Similarly, CNN-LSTM struggles with the lowest recall rates for Shell code (26.06%) and DoS (55.00%). In the class-wise analysis:
  - Reconnaissance: All models demonstrate robust performance, albeit with slight disparities, with the proposed CNN-BiLSTM model showcasing the highest recall (97.98%).
  - Exploits: Both CNN and the proposed CNN-BiLSTM model excel in identifying exploits, achieving the highest recall rates (92.26% and 93.40% respectively). Conversely, LeNet5 exhibits subpar performance in this class with 70.69%.
  - Normal Traffic: CNN-LSTM and the proposed CNN-BiLSTM model outperform others in accurately classifying normal traffic, achieving recall rates of 91.88% and 91.47% respectively. Notably, CNN displays the lowest recall for normal traffic classification with 45.89%.
  - Worms: AlexNet and the proposed CNN-BiLSTM model achieve the highest recall with 92.99% and 90.72% respectively. For the lowest recall, we have Lenet5 and CNN models with 72.68% and 73.76%.

- Backdoor: The alexNet model has the highest recall with 82.36% and the lowest we have CNN model with 33.45%. The proposed model achieves 49.29%.
- Fuzzers: The proposed model achieves the highest recall in this class with 68.69% and the lowest is the LeNet5 model with 47.89%.

- Precision: The detailed examination of model performance across specific attack classes unveils distinctive strengths and weaknesses, providing valuable insights into their effectiveness in intrusion detection.
  - Normal class: Both the proposed CNN-BiLSTM and LeNet5 models exhibit exceptional precision (>90%) in accurately identifying normal network traffic, reflecting their robust capability in distinguishing benign activities.
  - Reconnaissance & Exploits: CNN and the proposed CNN-BiLSTM model demonstrate superior precision (>80%) in detecting reconnaissance and exploit activities, highlighting their effectiveness in identifying potential malicious reconnaissance efforts and exploitation attempts.
  - Shellcode & Worms: CNN achieves notable precision rates exceeding 65.45% for shellcode and over 98% for worms, indicating its prowess in accurately classifying these specific attack types, showcasing its effectiveness in discerning sophisticated shellcode payloads and worm propagation attempts.
  - DoS: Despite its high overall accuracy, the AlexNet model displays highest precision (77.15%) in detecting denial-of-service attacks compared to CNN (48%), suggesting a potential area for improvement in accurately identifying DoS activities.
  - Backdoor, Analysis, Fuzzers: Precision rates across all models are relatively lower for these classes, indicating the higher complexity involved in accurately classifying these attack types. Notably, LeNet5 demonstrates superior performance with precision rates of 75% for backdoor and 52% for analysis classes. In contrast, CNN may not be optimal for these classes due to its lower precision, underscoring the challenges associated with effectively detecting backdoor, analysis, and fuzzing activities.

- Figure 4.4 shown the F1-score of all models for multi-class classification on UNSW-NB15. In our comprehensive evaluation of model performance for intrusion detection, several key observations and insights have emerged:

  - Challenges with Certain Classes: Most models encounter difficulties in accurately identifying the "Backdoor" and "Fuzzers" classes, reflected in their low F1-scores. This suggests the need for enhanced detection techniques or additional training data to improve performance in detecting these types of malicious activities.

  - Strong Performance in Normal Traffic Classification: Notably, all models exhibit consistently high F1-scores for classifying "Normal" traffic, with LeNet5 and the proposed CNN-BiLSTM model demonstrating the most robust performance in this regard.

  - Effective Detection of Exploits and Worms: The "Exploits" and "Worms" classes are well-identified by most models, with F1-scores exceeding 70%, indicating their effectiveness in detecting these types of attacks.

Class-wise Breakdown:

- Normal class: The proposed CNN-BiLSTM and CNN-LSTM models outshine others, achieving F1-scores surpassing 90%.
- Reconnaissance: All models perform commendably, with the proposed CNN-BiLSTM achieving the highest F1-score of 94.01%.
- Exploits & Worms: Both the proposed CNN-BiLSTM and CNN demonstrate superior performance, indicating robust detection capabilities.
- DoS: AlexNet exhibits the best performance with a score of 88.69%, though there is room for improvement across all models in this class.
- Shellcode & Analysis: AlexNet excels in detecting "Shellcode" (70.7%), and it performs best for "Analysis" (77.48%).
- Backdoor & Fuzzers: All models struggle significantly with these classes, with F1-scores below 50% in most cases.

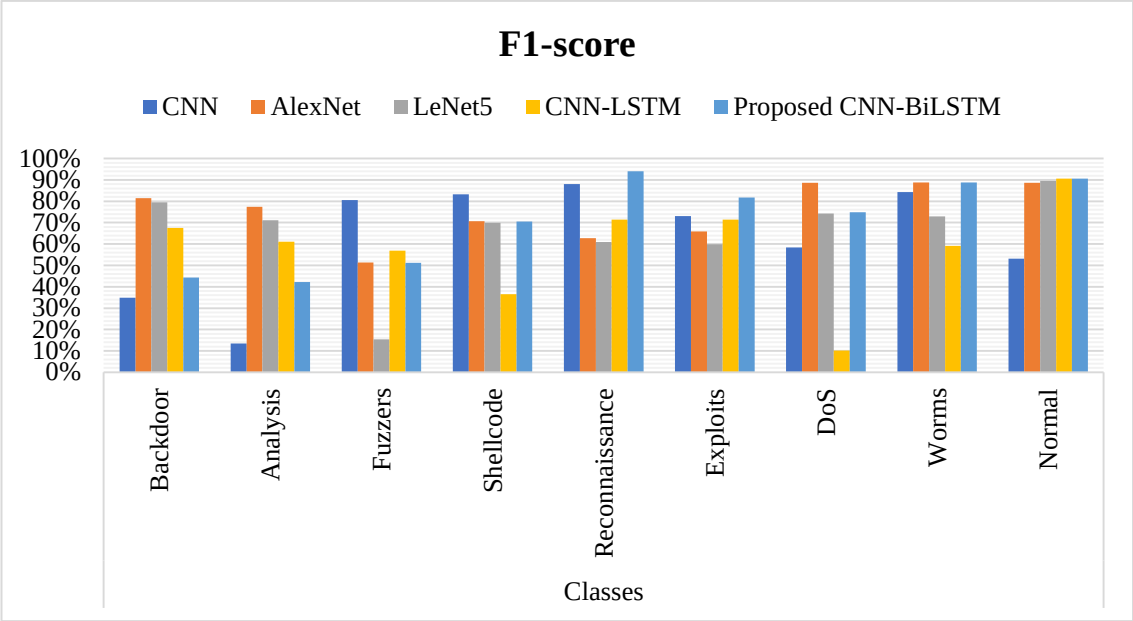

Figure 4.4 All models of multi-class classification on the UNSW-NB15 dataset were scored using the F1-score metric

Overall, the outcomes show that the proposed CNN-BiLSTM model is a promising approach for attack detection. It can effectively detect most types of attacks with high accuracy.

4.1.3 NSL-KDD dataset results

Table 4.3 shows the performance of different models on five different classes of network traffic: U2R, R2L, Probe, DoS, and Normal. The models are examined using different measures, including F1-score, recall, precision, and accuracy.

- Figure 4.5 depicts the accuracy of all models. The Proposed CNN-BiLSTM model achieves the highest accuracy (98.42%) followed by CNN-LSTM (98.38%) and AlexNet (97.85%). LeNet5 (97.31%) and CNN (79.14%) have lower accuracy.
- The evaluation of recall performance across different attack classes reveals notable distinctions among the models. Here's a detailed breakdown:

- U2R Class: The Proposed CNN-BiLSTM model achieves the highest recall (100%), indicating its ability to accurately capture all instances of the U2R class. Other models also exhibit strong performance in this class.
- R2L Class: All models, except AlexNet, demonstrate similar recall performance (around 70%) for this class. AlexNet exhibits the lowest recall (36.19%), suggesting challenges in accurately identifying instances of R2L attacks.
- Probe Class: All models exhibit good recall (above 91%) for this class, indicating their effectiveness in detecting probing activities.
- DoS Class: AlexNet, LeNet5, CNN-LSTM and the proposed CNN-BiLSTM models achieve very high recall (above 99%) for this class, highlighting their proficiency in identifying DoS. CNN model achieves the lowest recall with 83.07%.
- Normal Class: Similarly, all models demonstrate very high recall (above 98%) for normal traffic classification, indicating their effectiveness in distinguishing normal network behavior except CNN achieves 78.47%.

Overall, the Proposed CNN-BiLSTM model stands out with the best recall performance, achieving perfect recall for the U2R class. In contrast, AlexNet exhibits the lowest recall performance for the R2L class. These findings emphasize the robustness of the Proposed CNN-BiLSTM model in accurately identifying various types of attacks, positioning it as a strong candidate for intrusion detection tasks. Conversely, the lower recall performance of AlexNet underscores the need for further optimization in accurately detecting R2L.

- The precision analysis of various models for different attack classes underscores the superior performance of the Proposed CNN-BiLSTM model across most categories, with the exception of R2L. Here's a detailed breakdown of precision for each model across classes:
  - U2R: The Proposed CNN-BiLSTM model demonstrates the highest precision (99.31%), followed by CNN, LeNet5, CNN-LSTM, and AlexNet.

- R2L: LeNet5 and CNN-LSTM achieve the second-highest precision (94.98% and 94.57% respectively), surpassing the proposed CNN-BiLSTM model. AlexNet and CNN trail behind in precision for this class.
- Probe: AlexNet achieves the highest precision (99.10%), with the proposed CNN-BiLSTM model following closely. LeNet5, CNN-LSTM, and CNN exhibit slightly lower precision values.
- DoS: The AlexNet model attains the highest precision (99.89%), closely followed by the proposed CNN-BiLSTM, CNN-LSTM, and LeNet5. The CNN demonstrate lower precision for this class.
- Normal: The CNN-LSTM model maintains the highest precision (98.50%), with LeNet5 and the proposed CNN-BiLSTM model following suit. AlexNet and CNN exhibit lower precision values for normal traffic classification.

- F1-score: The performance evaluation of various models in the context of intrusion detection reveals notable distinctions in their F1-score metrics across different attack classes. Particularly, the proposed CNN-BiLSTM model emerges as the top performer across the majority of classes, showcasing its efficacy in accurately classifying instances of malicious activity.
    - For the U2R class, the Proposed CNN-BiLSTM model achieves a perfect F1-score of 99.65%, indicating flawless classification of all instances within this category. In contrast, other models trail behind, with CNN achieving an F1-score of 95.92%, LeNet5 at 89.92%, CNN-LSTM at 95.30%, and AlexNet at 88.68%.

    Similarly, across other classes:

    - R2L: CNN model achieves the highest F1 score with 91.35%, and the proposed CNN-BiLSTM model secures the second highest F1-score (81.97%), closely trailing behind CNN-LSTM (79.77%). AlexNet and LeNet5 exhibit lower F1-scores at 53.14% and 68.82% respectively.
    - Probe: Once again, the Proposed AlexNet model leads with the highest F1-score (98.28%), followed by CNN-LSTM, the proposed CNN-BiLSTM mode, and AlexNet.

- DoS: The AlexNet model attains the highest F1-score (99.82%), followed closely by the proposed CNN-BiLSTM, CNN-LSTM, LeNet5, and CNN.
- Normal: The CNN-LSTM model outperforms others with the highest F1-score (98.81%), with the proposed CNN-BiLSTM, LeNet5, AlexNet, and CNN following suit.

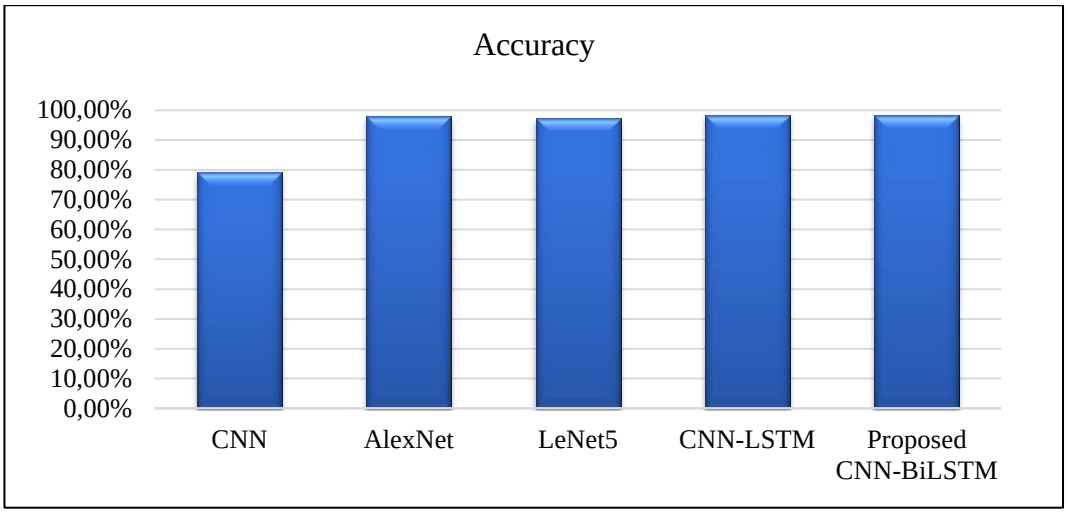

Figure 4.5 Results of multiclass classification accuracy for various models on the NSL-KDD dataset

Table 4.3 The multi-class classification results using NSL-KDD

Models	Metrics (%)	Class				
		U2R	R2L	Probe	DoS	Normal
CNN	Accuracy	**79.14**				
	Recall	96.32	84.36	86.23	83.07	78.47
	Precision	89.84	88.25	93.34	83.39	52.16
	F1-Score	95.92	91.35	90.54	83.21	68.56
AlexNet	Accuracy	**97.85**				
	Recall	86.67	36.19	97.40	99.75	99.57
	Precision	81.18	38.25	99.10	99.89	96.81
	F1-Score	88.68	53.14	98.28	99.82	98.17
LeNet5	Accuracy	**97.31**				
	Recall	94.86	53.96	96.16	99.44	98.14
	Precision	88.84	94.98	97.13	99.07	98.03
	F1-Score	89.92	68.82	96.64	99.26	98.08
CNN-LSTM		**98.38**				
	Recall	96.67	68.97	97.40	99.89	99.12
	Precision	96.98	94.57	98.74	99.48	98.50
	F1-Score	95.30	79.77	98.07	99.68	98.81
Proposed CNN-BiLSTM	Accuracy	**98.42**				
	Recall	100	74.31	91.05	99.69	99.02
	Precision	99.31	91.39	93.54	99.63	98.35
	F1-Score	99.65	81.97	91.79	99.66	98.69

4.1.4 Model benchmark

An improved level of accuracy was achieved with the proposed model. Based on Table 4.4, our model had an accuracy of 98.42%, which was the highest among comparable studies.

Table 4.4 Benchmark results of the model

References	Method used	Accuracy for multiclass classification on average		
		InSDN dataset	UNSW-NB15	NSL-KDD
(Sathya & Thangarajan, 2015)	Deep Neural Network (DNN)	-	-	91.10%
(Tang et al., 2016)	J48 which is based on the decision tree	-	-	75.75%
(Jiang et al., 2020)	CNN-BiLSTM	-	77.16%	83.58%
(Tang, Mhamdi, McLernon, Zaidi, & Ghogho, 2018)	GRU-RNN	-	-	89%
(Choobdar, Naderan, & Naderan, 2022)	Stacked Auto-Encoders	-	-	94.88%
This study	Proposed CNN-BiLSTM	97.12%	84.23%	98.42%

4.1.5 Summary for what was covered

We developed a solution for NID based on CNN and BiLSTM. The suggested model makes use of the integrity of CNN and BiLSTM to enhance detection capacity across a variety of datasets. Additionally, we demonstrated that our suggested technique outperformed either a single CNN, LeNet5, AlexNet, or CNN-LSTM in terms of accuracy by 97.12 % in the InSDN dataset, 84.23% in UNSW-NB15 and 98.42% in

NSL-KDD dataset. Furthermore, by balancing the dataset using the random over-sampling approach and selecting features using an RF classifier along with the RFE approach, the detection model's performance was improved.

4.2 Attention-based CNN-BILSTM DL Approach for NIDS in SDN

4.2.1 InSDN dataset results

Table 4.5 provides detailed findings for multi-class classification using several classifiers on the InSDN dataset. In terms of different metrics, the performance of various attacks varies significantly amongst classifiers.

Table 4.5 InSDN Multi-class classification results

Models	**Metrics (%)**	**Class**								
		U2R	BFA	DDoS	DoS	BOTNET	Normal	R2L	Probe	WEB-ATTACK
CNN	Accuracy	**83.86**								
	Recall	19.30	100	89.85	99.98	12.50	84.11	98.26	44.26	98.41
	Precision	39.34	95.92	99.99	86.65	11.11	99.49	69.35	67.37	6.18
	F1-Score	25.90	97.92	97.92	94.65	11.76	91.16	81.31	53.42	11.63
AlexNet	Accuracy	**83.95**								
	Recall	53.02	74.29	70	100	50.59	98.45	77.58	85.97	27.14
	Precision	92.35	100	100	68	33.15	95.69	98.32	67.25	01.26
	F1-Score	67.24	67.84	82	81	3.25	96.24	86.47	75.36	03.48
LeNet5	Accuracy	**90.49**								
	Recall	75.58	100	90.13	86.92	62.50	72.38	98.60	91.01	98.41
	Precision	05.63	100	99.99	86.92	15.62	95.88	98.98	98.01	8.21
	F1-Score	10.48	100	94.80	93.00	25.00	82.49	98.79	94.38	15.16
CNN-LSTM	Accuracy	**93.05**								
	Recall	72.25	94.78	98.24	91.26	5.10	67.56	100	98.57	91.55
	Precision	61.58	100	94.36	100	62.50	100	86.35	43.25	88.89
	F1score	66.47	97.12	96.52	95.79	9.43	80.69	92.47	60.18	83.04
Proposed Attention based CNN-BiLSTM	Accuracy	**98.03**								
	Recall	63.58	100	100	100	62.38	98.37	97.56	99.07	92.00
	Precision	57.23	89.04	96.4	92.45	10.25	99.19	100	98.52	16.00
	F1-Score	60.87	94.74	98.71	96.57	17.47	99.85	98.41	98.41	27.00

- Figure 4.6 shows the accuracy of all models. All models in the analysis demonstrate strong accuracy, ranging from 83.86% to 98.03%. This metric reflects the overall correctness of the models' predictions across all classes. The high accuracy values indicate that the models perform well in making correct

predictions for the majority of instances in the dataset. Notably, the proposed attention-based CNN-BiLSTM model outcomes the highest accuracy of 98.03%, showcasing its effectiveness in accurately classifying network traffic across various classes. The CNN model achieves the lowest accuracy with 83.86%. For AlexNet, LeNet5 and CNN-LSTM models have 83.95%, 90.49%, and 93.05% respectively.

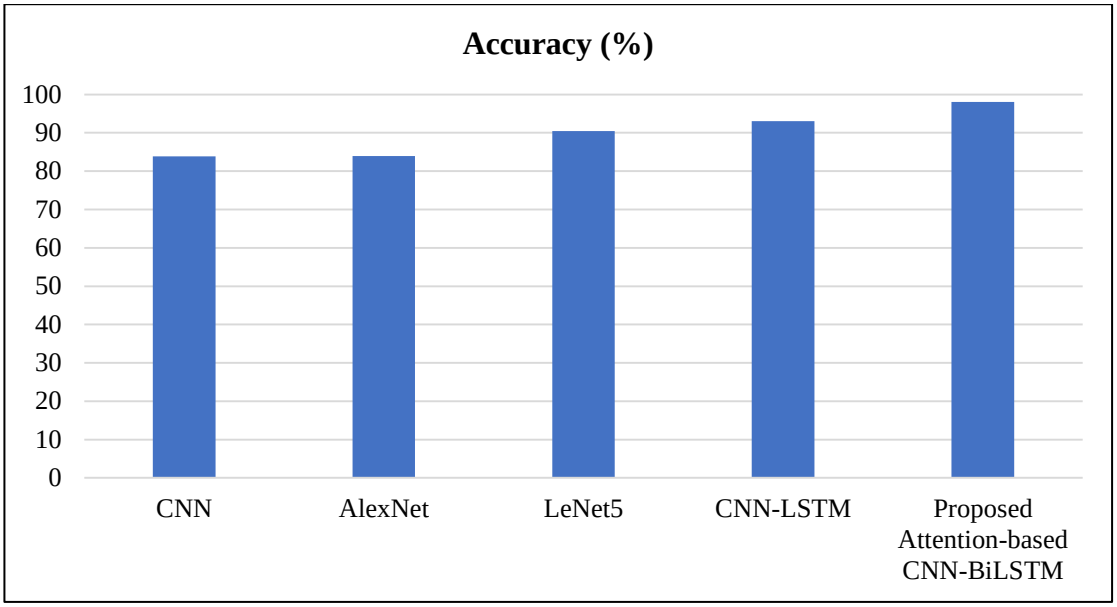

Figure 4.6 Results of multiclass classification accuracy for various models on the InSDN dataset

- Precision: The performance of various models in classifying different attack types varies significantly.

  - Specifically, precision levels for the U2R class exhibit considerable variability, spanning from a mere 5.63% to 92.35%. This inconsistency implies a tendency among the models to frequently misclassify other classes as U2R, thereby generating a considerable number of false positives.

  - Conversely, the Normal and DoS classes generally boast high precision rates, with 100% by the CNN-LSTM model. Such high precision values indicate the models' proficiency in accurately discerning these classes.

- For the BOTENT class has the highest precision by the CNN-LSTM model with 62.50% followed by Alexnet with 33.15%. The lowest precision model is the CNN-BiLSTM model with 10.25%.

- Notably, the BFA, DDoS, and Normal classes consistently demonstrate superior precision across all models, suggesting the models' adeptness in identifying these classes with minimal false positives. However, the WEB-ATTACK class presents a notable challenge, with consistently low precision across all models except the CNN-LSTM model. This recurring issue implies a frequent misclassification of other classes as WEB-ATTACK, consequently, resulting in a high incidence of false positives.

➢ The recall: Across the spectrum of model performance, certain trends and challenges emerge, shedding light on their efficacy in classifying various attack types. Notably, all models encounter difficulty in recalling the BOTNET class, with consistently low values across most cases. Conversely, the Normal class boasts generally high recall rates across models, albeit exceptions exist, notably with CNN and CNN-LSTM. Additionally, R2L and Probe classes demonstrate robust recall in most models, indicative of their distinguishability.

Delving into specific model observations, the CNN model, grapples with low recall for several classes, notably U2R (19.30%) and BOTNET (12.50%). AlexNet, while improving upon CNN's performance in certain areas, still faces challenges with U2R (53.02%), DDoS (70%), BOTNET (50.59%), and WEB-ATTACK (27.14%) classifications. LeNet5 emerges as a frontrunner, exhibiting superior recall across most classes compared to CNN and AlexNet, albeit it encounters issues with BOTNET recall. CNN-LSTM, although showcasing decent recall for select classes, falls short in U2R, DoS, BOTNET, and WEB-ATTACK classifications.

On the other hand, the proposed Attention-based CNN-BiLSTM model garners attention for its high overall accuracy and recall rates, particularly excelling in BFA (100%), DoS (100%), and DDoS (100%) classifications. However, similar to

other models, it faces challenges in achieving optimal recall for the BOTNET (62.38%) class, suggesting room for improvement in this regard.

- Figure 4.7 depicts the F1-score results. The analysis of model performance reveals notable trends and challenges in classifying different types of network traffic.

  - Across all models, the BOTNET class poses a significant challenge, with F1-scores ranging from 3.25% to 25.00%, indicating difficulties in accurately identifying this particular attack type.

  - Normal class exhibits consistent and high F1-scores, surpassing 80% across all models, while R2L and Probe classes also demonstrate respectable performance, with F1-scores above 75% in multiple instances.

  - The performance of U2R, DoS, and WEB-ATTACK classes varies across models. Notably, CNN-LSTM and the Proposed Attention-based CNN-BiLSTM models generally achieve higher F1-scores for these classes compared to others, suggesting their efficacy in handling these specific attack types.

  - Examining model-specific observations, CNN stands out for its good overall accuracy (83.86%), yet it struggles with relatively low F1-scores for several classes, particularly U2R (25.90%), BOTNET (11.76%), and WEB-ATTACK (11.63), indicating a trade-off between overall accuracy and class-specific performance.

  - In contrast, AlexNet excels in the Normal class (96.24%) but faces challenges with most other classes, especially BOTNET (3.25%). Despite its lower overall accuracy.

  - LeNet5 achieves a high F1-score for the BFA class (100%) and performs well for DoS (93.00%) and Probe (94.38%) classes compared to other models.

  - The CNN-LSTM model strikes a balance between overall accuracy (93.05%) and class-specific F1-scores, performing admirably for BFA (97.12%), DDoS (96.52%), and R2L (92.47%) classes. The CNN-LSTM model achieves the lowest F1-score with the BOTNET class (9.43%).

- The proposed Attention-based CNN-BiLSTM model emerges as the top performer, boasting the highest overall accuracy (98.03%) and strong F1-scores for most classes, including Normal (99.85%), DDoS (98.71%), DoS (96.57%), and R2L (98.41%), underscoring its effectiveness in handling a diverse range of attack types except the BOTNET (17.47%), and the WEB ATTACK (27.00%) attacks.

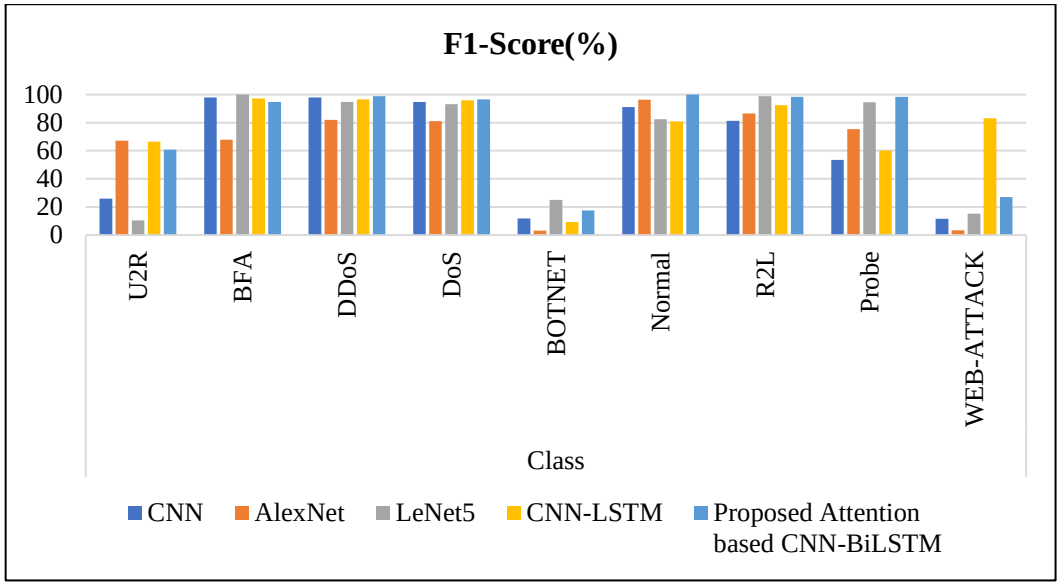

Figure 4.7 Results of multiclass classification F1-score for various models on the InSDN dataset

4.2.2 NSL-KDD dataset results

Table 4.6 shows the performance of different models on five different classes of network traffic: U2R, R2L, Probe, DoS, and Normal. The models are evaluated using different measures, including accuracy, F1-score, precision, and recall.

Table 4.6 NSL KDD Multi-class classification results

Models	**Metrics (%)**	**Class**				
		U2R	**R2L**	**Probe**	**DoS**	**Normal**
CNN	Accuracy			**97.78**		
	Recall	86.67	60.73	97.54	99.97	98.40
	Precision	09.59	93.75	95.94	99.49	98.25
	F1-Score	17.28	73.74	96.73	99.60	98.32
AlexNet	Accuracy			**99.14**		
	Recall	66.67	90.15	99.00	99.94	99.09
	Precision	25.97	89.90	98.97	99.77	99.42
	F1-Score	37.38	90.02	98.99	99.85	99.26

Table 4.6 NSL KDD Multi-class classification results (continue)

Models	**Metrics (%)**	**Class**				
		U2R	**R2L**	**Probe**	**DoS**	**Normal**
LeNet5	Accuracy	**97.18**				
	Recall	86.67	60.22	97.57	99.90	96.99
	Precision	05.54	96.18	95.00	98.56	98.52
	F1-Score	10.42	74.07	96.27	99.23	97.75
CNN-LSTM		**98.41**				
	Recall	80.00	96.41	99.19	99.86	99.41
	Precision	21.05	71.32	97.54	99.76	99.78
	F1-Score	33.33	81.99	98.35	99.81	98.58
Proposed attention-based CNN-BiLSTM	Accuracy	**99.42**				
	Recall	83.33	97.42	99.28	99.69	99.69
	Precision	32.89	71.68	97.93	99.58	99.79
	F1-Score	47.17	82.59	98.60	99.66	99.63

- Figure 4.8 shows the accuracy of all models. Among the evaluated models, the proposed attention-based CNN-BiLSTM stands out as the top performer, achieving the highest accuracy of 99.42% and excelling in most other metrics, including recall, precision, and F1-score. Following closely, AlexNet demonstrates strong performance with an accuracy of 99.14%, particularly notable for its precision and F1-score, although its recall for certain classes lags behind the attention-based CNN-BiLSTM model. CNN-LSTM also showcases commendable performance with an accuracy of 98.41% and strong recall and F1-score across most classes, albeit with lower precision compared to some other models. LeNet5 and CNN exhibit similar accuracies around 97% and performance in other metrics, making them viable options for applications prioritizing factors like computational efficiency over absolute accuracy. Ultimately, while the Proposed attention-based CNN-BiLSTM emerges as the optimal choice based on the provided data, selecting the most suitable model for specific applications necessitates considering individual requirements and priorities.

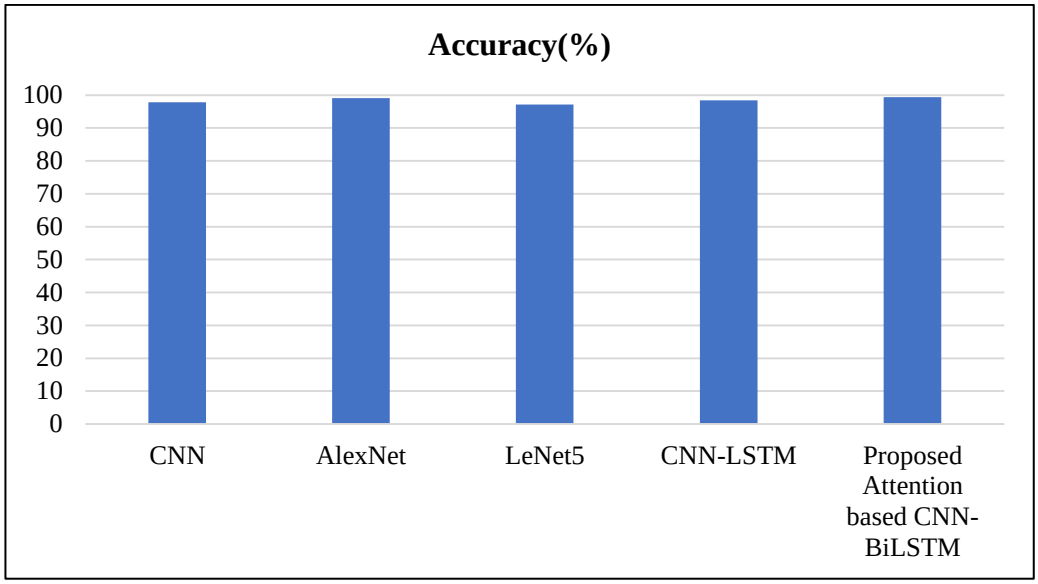

Figure 4.8 Results of multiclass classification accuracy for various models on the NSL-KDD dataset.

- Precision: The precision analysis highlights key insights into the performance of the evaluated models across various attack classes.

  - Specifically, the proposed attention-based CNN-BiLSTM model demonstrates the highest precision for the Normal class (99.79%), showcasing its proficiency in accurately identifying normal network traffic. Following closely, AlexNet and CNN-LSTM also exhibit high precision for the Normal class, further solidifying their capability in this regard.

  - For the U2R class, the proposed attention-based CNN-BiLSTM model emerges with the highest precision (32.89%), indicating its effectiveness in correctly classifying U2R attacks. However, it's notable that precision for the U2R class remains relatively low across all models.

  - The proposed attention-based CNN-BiLSTM model displays impressive precision for the Probe class (97.93%) and DoS class (99.58%), underscoring its robustness in identifying probing and DoS attacks. Nonetheless, its precision for the R2L class falls short compared to LeNet5, suggesting room for improvement in classifying R2L attacks. Overall, while the proposed attention-based CNN-BiLSTM model demonstrates strong precision across several classes.

- Recall: The evaluation of model recall across different attack classes reveals significant variations in performance.

  - For the U2R class, the proposed attention-based CNN-BiLSTM model achieves the recall with 83.33%, closely followed by CNN and LeNet5 (86.67%), while AlexNet lags with the lowest recall (66.67%).

  - In contrast, for the R2L class, the proposed attention-based CNN-BiLSTM model demonstrates the highest recall (97.42%), followed by CNN-LSTM (96.41%), with AlexNet showing the second-highest recall (90.15%), and CNN and LeNet5 exhibiting the lowest recall (around 60%).

  - For the Probe class, all models achieve exceptionally high recall rates (above 97%), with the proposed attention-based CNN-BiLSTM model leading with the highest recall (99.28%). Similarly, for the DoS class, all models achieve recall rates exceeding 99%, with CNN recording the highest recall (99.97%).

  - In terms of the Normal class, all models exhibit very high recall rates (above 96%), with both the proposed attention-based CNN-BiLSTM model and CNN-LSTM model showcasing the highest recall (99.69%).

  Overall, the Proposed attention-based CNN-BiLSTM model demonstrates superior recall performance across most classes, followed by CNN-LSTM and CNN, while AlexNet and LeNet5.

- The assessment of F1-scores across different attack classes provides insights into the overall performance of the models. Notably, for the U2R class, the proposed attention-based CNN-BiLSTM model achieves the highest F1-score (47.17%), followed by AlexNet (37.38%) and CNN-LSTM (33.33%), indicating its effectiveness in balancing precision and recall for this class.

  -Similarly, for the R2L class, the AlexNet model leads with the highest F1-score (90.02%), closely followed by the proposed attention-based CNN-BiLSTM model (82.59%) and CNN-LSTM (81.99%), highlighting its robustness in capturing instances of R2L attacks.

-In terms of the Probe class, although AlexNet achieves the highest F1-score (98.99%), the proposed attention-based CNN-BiLSTM model secures the second-highest score (98.60%), demonstrating its proficiency in identifying probing activities.

-For the DoS class, all models exhibit very high F1-scores (> 99%), with the Proposed attention-based CNN-BiLSTM model securing the second-highest score (99.66%), reaffirming its capability in detecting DoS attacks.

- Similarly, for the Normal class, all models achieve very high F1-scores (> 98%), with the Proposed attention-based CNN-BiLSTM model again securing the second-highest score (99.63%), underscoring its reliability in discerning normal network traffic.

Overall, the proposed attention-based CNN-BiLSTM model showcases commendable performance across all classes, particularly excelling in the U2R and Normal classes. Figure 4.9 depicts the F1-score of all models as follows:

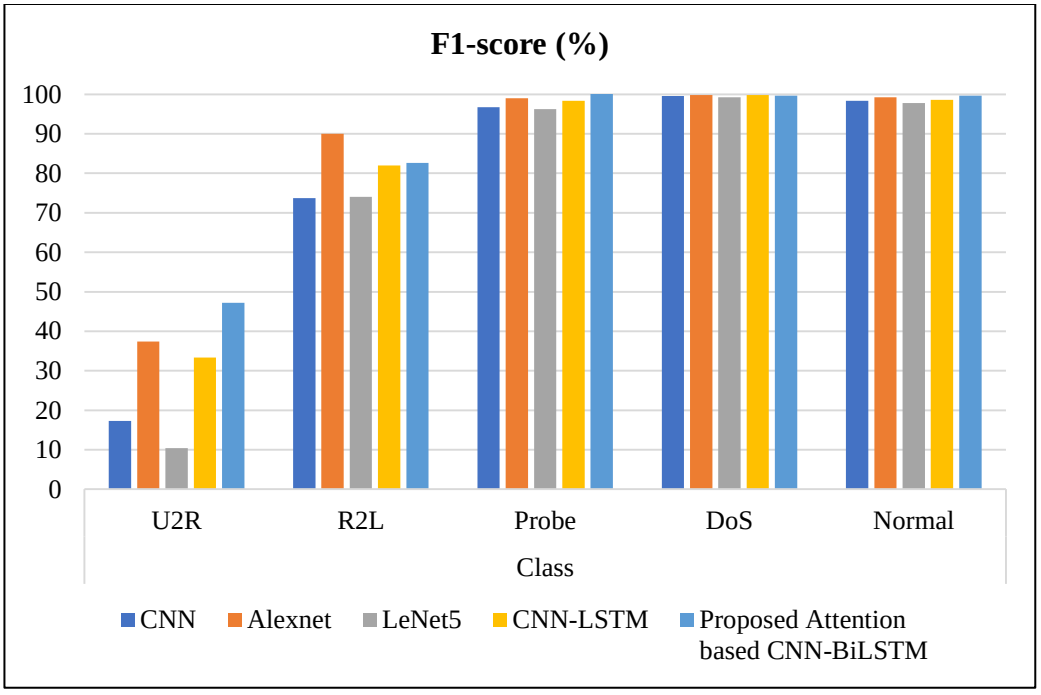

Figure 4.9 Results of multiclass classification F1-score for various models on the NSL-KDD dataset

4.2.3 Model benchmark

An improved level of accuracy was achieved with the proposed model. Based on Table 4.7, our model had an accuracy of 98.03% for the InSDN dataset and 99.42% for the NSL-KDD dataset, which was the highest among comparable studies.

Table 4.7 Benchmark results of the model

References	Method used	Accuracy for multiclass classification on average		
		InSDN dataset	UNSW-NB15	NSL-KDD
(Sathya & Thangarajan, 2015)	Deep Neural Network (DNN)	-	-	91.10%
(Tang et al., 2016)	J48 which is based on the decision tree	-	-	75.75%
(Jiang et al., 2020)	CNN-BiLSTM	-	77.16%	83.58%
(Tang et al., 2018)	GRU-RNN	-	-	89%
(Choobdar et al., 2022)	Stacked Auto-Encoders	-	-	94.88%
(Said, Sabir, & Askerzade, 2023)	CNN-BiLSTM	97.12%	84.23%	98.42%
This study	Attention-based CNN-BiLSTM	98.03%	-	99.42%

4.2.4 Summary of what was covered

We investigated the efficacy of various DL models for NIDS within the context of SDN. With a high detection rate of 98.03% for identifying attacks in the InSDN dataset and 99.42% in the NSL-KDD dataset, the suggested Attention-based CNN-BiLSTM model demonstrated the best accuracy. This emphasizes how important it is to incorporate attention mechanisms into the DL model so that the model can efficiently extract important characteristics from network traffic data. Additionally, the CNN-

LSTM model also demonstrated high performance, achieving an accuracy of 98.38% in the NSL-KDD dataset and 93.05 % in the InSDN dataset. This suggests that the combination of CNN and BiLSTM architectures effectively captures both local and global temporal dependencies in the network traffic data, resulting in accurate intrusion detection. In contrast, the CNN and AlexNet models fared less favorably, with accuracies of 83.86% and 83.95% respectively in the InSDN dataset but for the NSL-KDD dataset, CNN and AlexNet achieved 97.78% and 99.14%.

5. RESEARCH CONCLUSION AND FUTURE WORK

5.1 Conclusion

Based on the outcomes of this research study, it is evident that DL models play an important role in enhancing NIDS. In the first study, we proposed a novel solution leveraging a combination of CNN and BiLSTM architectures. Through comprehensive experimentation across multiple datasets including InSDN, UNSW-NB15, and NSL-KDD, our proposed model consistently outperformed individual models such as CNN, LeNet5, AlexNet, and CNN-LSTM, achieving remarkable accuracy rates of 97.12%, 84.23%, and 98.42% respectively. This improvement in accuracy was achieved by leveraging techniques such as random oversampling for dataset balancing and feature selection employing an RF classifier coupled with RFE.

In the second study, we delved deeper into exploring various deep learning models within the context of SDN for intrusion detection. Our attention-based CNN-BiLSTM model emerged as the top performer, achieving an impressive accuracy rate of 98.03% for the InSDN dataset and 99.42% for the NSL-KDD dataset. The incorporation of attention mechanisms into the deep learning framework proved to be pivotal in effectively capturing critical features in network traffic data, thereby enhancing the model's detection capabilities. Additionally, the CNN-LSTM model also demonstrated high performance, achieving accuracy rates of 93.05% and 98.38% for the InSDN and NSL-KDD datasets respectively. This underscores the effectiveness of combining CNN and BiLSTM architectures in capturing both local and global temporal dependencies in network traffic data.

However, it is important to note that while the CNN and AlexNet models showcased decent accuracy rates, they were outperformed by the Attention-based CNN-BiLSTM and CNN-LSTM models in terms of overall detection performance. These findings highlight the significance of leveraging advanced deep learning architectures and attention mechanisms for robust intrusion detection in network environments. Overall, our research contributes to the advancement of NIDS by presenting novel

methodologies and models that offer enhanced detection capabilities, thereby bolstering network security in SDN environments.

5.2 Future Work

This study accomplishes all its research objectives, notably enhancing the accuracy of NIDS. Nevertheless, for researchers seeking to delve deeper into NIDS through hybrid deep learning approaches, the following recommendations are provided:

1. Enhancing the process of feature selection within NIDS involves exploring alternative methodologies beyond traditional methods. This entails delving into heuristic and metaheuristic algorithms to refine the selection process, thus optimizing the choice of features for improved detection accuracy and efficiency. Leveraging these advanced algorithms within the framework of suggested models can offer novel insights and approaches, potentially leading to more robust and effective intrusion detection systems in diverse network environments.

2. The utilization of various techniques for balancing datasets, such as SMOTE, under-sampling, and others, holds significant importance in the context of NIDS. These techniques play an essential role in addressing the inherent class imbalance often observed in network traffic data, where instances of normal behavior significantly outnumber instances of malicious activity. By employing methods like SMOTE, which generates synthetic samples for the minority class, and under-sampling, which reduces the number of instances in the majority class, NIDS can better capture and learn from both normal and malicious network behaviors. Moreover, when combined with appropriate DL-based approaches like CNNs or RNNs, these balanced datasets contribute to the creation of more accurate and robust intrusion detection systems. This integration of data balancing techniques with advanced machine learning models enhances the detection capabilities of NIDS, thereby improving network security and resilience against potential cyber threats.

3. Exploring the potential of transfer and ensemble learning techniques to improve the performance of detection and classification of attacks.

4. Multi-modeling, also known as ensemble modeling or model fusion, is a technique in machine learning and data science where multiple models are combined to improve predictive performance or address different aspects of a problem. This approach leverages the diversity of multiple models to create a more robust and accurate prediction compared to using a single model alone.

5. Deploying the suggested model in a real SDN system involves implementing the hybrid model within the network infrastructure and conducting a comprehensive evaluation of its throughput and latency performance. This evaluation encompasses rigorous testing under various network conditions, assessing the model's ability to efficiently process and transmit data packets while minimizing delays and latency. Additionally, it involves monitoring the network's overall performance metrics, such as bandwidth utilization, packet loss rates, and responsiveness, to ensure that the deployed model effectively meets the system's requirements and objectives. Through thorough evaluation and analysis, any potential bottlenecks or limitations in the model's performance can be identified and addressed, facilitating the optimization and enhancement of the SDN system's overall efficiency and effectiveness.
6. Developing a NIDPS based on an analysis of network traffic and network host logs to gain a comprehensive understanding of potential attacks in SDN.

REFERENCES

Albahar, M. A. (2019). Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments. *Security and Communication Networks, 2019*, 1-9.

Alshamrani, A., Chowdhary, A., Pisharody, S., Lu, D., & Huang, D. (2017). *A defense system for defeating DDoS attacks in SDN based networks.* Paper presented at the Proceedings of the 15th ACM international symposium on mobility management and wireless access.

Alsharaiah, M., Abualhaj, M., Baniata, L., Al-saaidah, A., Kharma, Q., & Al-Zyoud, M. (2024). An innovative network intrusion detection system (NIDS): Hierarchical deep learning model based on Unsw-Nb15 dataset. *International Journal of Data and Network Science, 8*(2), 709-722.

Amudha, P., & Rauf, H. A. (2011). *Performance analysis of data mining approaches in intrusion detection.* Paper presented at the 2011 International Conference on Process Automation, Control and Computing.

Antonucci, D. (2017). *The cyber risk handbook: Creating and measuring effective cybersecurity capabilities*: John Wiley & Sons.

Azodolmolky, S. (2013). *Software defined networking with OpenFlow*: Packt Publishing.

Bakhshi, T., & Ghita, B. (2021). Anomaly detection in encrypted internet traffic using hybrid deep learning. *Security and Communication Networks, 2021*, 1-16.

Bian, L., Zhang, L., Zhao, K., Wang, H., & Gong, S. (2021). Image-Based Scam Detection Method Using an Attention Capsule Network. *IEEE Access, 9*, 33654-33665.

Bisong, E. (2019). *Building machine learning and deep learning models on Google cloud platform*: Springer.

Blenk, A., Basta, A., Reisslein, M., & Kellerer, W. (2015). Survey on network virtualization hypervisors for software defined networking. *IEEE communications surveys & tutorials, 18*(1), 655-685.

Bonaccorso, G. (2020). *Mastering Machine Learning Algorithms: Expert techniques for implementing popular machine learning algorithms, fine-tuning your models, and understanding how they work*: Packt Publishing Ltd.

BOUKRIA, S., & GUERROUMI, M. (2019). *Intrusion detection system for SDN network using deep learning approach.* Paper presented at the 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS).

Breiman, L. (2001). Random forests. *Machine learning, 45*, 5-32.

Brownlee, J. (2016). *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*: Machine Learning Mastery.

Buyya, R., & Son, J. (2018). *Software-defined multi-cloud computing: a vision, architectural elements, and future directions.* Paper presented at the Computational Science and Its Applications–ICCSA 2018: 18th International Conference, Melbourne, VIC, Australia, July 2-5, 2018, Proceedings, Part I 18.

Cao, K., Zhu, J., Feng, W., Ma, C., Liu, M., & Du, T. (2021). *Network intrusion detection based on dense dilated convolutions and attention mechanism.* Paper presented at the 2021 International Wireless Communications and Mobile Computing (IWCMC).

Choobdar, P., Naderan, M., & Naderan, M. (2022). Detection and multi-class classification of intrusion in software defined networks using stacked auto-encoders and CICIDS2017 dataset. *Wireless Personal Communications*, 1-35.

Coburn, A., Daffron, J., Quantrill, K., Leverett, E., Bordeau, J., Smith, A., & Harvey, T. (2019). Cyber risk outlook. *Centre for Risk Studies, University of Cambridge, in collaboration with Risk Management Solutions, Inc*.

Dawoud, A., Shahristani, S., & Raun, C. (2018). Deep learning and software-defined networks: Towards secure IoT architecture. *Internet of Things, 3*, 82-89.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on software engineering*(2), 222-232.

Dhanabal, L., & Shantharajah, S. (2015). A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering, 4*(6), 446-452.

Doherty, J. (2016). *SDN and NFV simplified: a visual guide to understanding software defined networks and network function virtualization*: Addison-Wesley Professional.

Duan, Q., & Toy, M. (2016). *Virtualized software-defined networks and services*: Artech House.

Elsayed, M. S., Le-Khac, N.-A., & Jurcut, A. D. (2020). InSDN: A Novel SDN Intrusion Dataset. *IEEE Access, 8*, 165263-165284. doi:10.1109/access.2020.3022633

Elsayed, M. S., Le-Khac, N. A., & Jurcut, A. D. (2020). InSDN: A Novel SDN Intrusion Dataset. *IEEE Access, 8*, 165263-165284. doi:10.1109/ACCESS.2020.3022633

Fellin, C., & Haney, M. (2014, 27 June-2 July 2014). *Preventing the Mistraining of Anomaly-Based IDSs through Ensemble Systems.* Paper presented at the 2014 IEEE World Congress on Services.

Géron, A. (2019). Hands-on machine learning with Scikit-Learn. *Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems, 1*.

Guan, X., Fan, Y., Qin, Q., Deng, K., & Yang, G. (2020). Construction of science and technology achievement transfer and transformation platform based on deep learning and data mining technology. *Journal of Intelligent & Fuzzy Systems, 39*(2), 1843-1854.

Hoang, D. B., & Pham, M. (2015). *On software-defined networking and the design of SDN controllers.* Paper presented at the 2015 6th International Conference on the Network of the Future (NOF).

Hu, H., Han, W., Ahn, G.-J., & Zhao, Z. (2014). *FLOWGUARD: Building robust firewalls for software-defined networks.* Paper presented at the Proceedings of the third workshop on Hot topics in software defined networking.

Huang, D., Chowdhary, A., & Pisharody, S. (2018). *Software-Defined networking and security: from theory to practice*: CRC Press.

Jackson, E., & Agrawal, R. (2019). *Performance evaluation of different feature encoding schemes on cybersecurity logs*: IEEE.

Jacob, N. M., & Wanjala, M. Y. (2017). A review of intrusion detection systems. *Global Journal of Computer Science and Information Technology Research. Framingham: Global Journals Inc, 5*(4), 1-5.

Jain, R., & Paul, S. (2013). Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine, 51*(11), 24-31.

Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network. *IEEE Access, 8*, 32464-32476. doi:10.1109/ACCESS.2020.2973730

Jie, L., & Chengzhi, W. (2019, 2019/04). *Feature Selection and Deep Learning based Approach for Network Intrusion Detection.* Paper presented at the Proceedings of the 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019).

Khan, F. A., Gumaei, A., Derhab, A., & Hussain, A. (2019). A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access, 7*, 30373-30385.

Klöti, R., Kotronis, V., & Smith, P. (2013). *OpenFlow: A security analysis.* Paper presented at the 2013 21st IEEE International Conference on Network Protocols (ICNP).

Kokila, R., Selvi, S. T., & Govindarajan, K. (2014). *DDoS detection and analysis in SDN-based environment using support vector machine classifier*. Paper presented at the 2014 sixth international conference on advanced computing (ICoAC).

Kreutz, D., Ramos, F. M., & Verissimo, P. (2013). *Towards secure and dependable software-defined networks.* Paper presented at the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE, 103*(1), 14-76.

Laghrissi, F., Douzi, S., Douzi, K., & Hssina, B. (2021). IDS-attention: an efficient algorithm for intrusion detection systems using attention mechanism. *Journal of Big Data, 8*(1), 149. doi:10.1186/s40537-021-00544-5

Lan, M., Luo, J., Chai, S., Chai, R., Zhang, C., & Zhang, B. (2020). *A novel industrial intrusion detection method based on threshold-optimized CNN-BiLSTM-Attention using ROC curve.* Paper presented at the 2020 39th Chinese Control Conference (CCC).

Le, A., Dinh, P., Le, H., & Tran, N. C. (2015). *Flexible network-based intrusion detection and prevention system on software-defined networks.* Paper presented at the 2015 International Conference on Advanced Computing and Applications (ACOMP).

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems, 2*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.

Li, H., Wei, F., & Hu, H. (2019). *Enabling dynamic network access control with anomaly-based IDS and SDN.* Paper presented at the Proceedings of the ACM international workshop on security in software defined networks & network function virtualization.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM computing surveys (CSUR), 50*(6), 1-45.

Liyanage, M., Ylianttila, M., & Gurtov, A. (2014). *Securing the control channel of software-defined mobile networks.* Paper presented at the Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014.

Malaiya, R. K., Kwon, D., Kim, J., Suh, S. C., Kim, H., & Kim, I. (2018). *An empirical evaluation of deep learning for network anomaly detection.* Paper presented at the 2018 International Conference on Computing, Networking and Communications (ICNC).

Manso, P., Moura, J., & Serrão, C. (2019). SDN-based intrusion detection system for early detection and mitigation of DDoS attacks. *Information, 10*(3), 106.

Matsumoto, S., Hitz, S., & Perrig, A. (2014). *Fleet: Defending SDNs from malicious administrators.* Paper presented at the Proceedings of the third workshop on Hot topics in software defined networking.

Mehdi, S. A., Khalid, J., & Khayam, S. A. (2011). *Revisiting traffic anomaly detection using software defined networking.* Paper presented at the Recent Advances in Intrusion Detection: 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings 14.

Mousavi, S. M., & St-Hilaire, M. (2018). Early detection of DDoS attacks against software defined network controllers. *Journal of Network and Systems Management, 26*, 573-591.

Moustafa, N., & Slay, J. (2015). *UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).* Paper presented at the 2015 military communications and information systems conference (MilCIS).

Mueller, J. P., & Massaron, L. (2021). *Machine learning for dummies*: John Wiley & Sons.

Nanda, S., Zafari, F., DeCusatis, C., Wedaa, E., & Yang, B. (2016). *Predicting network attack patterns in SDN using machine learning approach.* Paper presented at the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN).

Narayanadoss, A. R., Truong-Huu, T., Mohan, P. M., & Gurusamy, M. (2019). *Crossfire attack detection using deep learning in software defined its networks.* Paper presented at the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring).

Niyaz, Q., Sun, W., & Javaid, A. Y. (2016). A deep learning based DDoS detection system in software-defined networking (SDN). *arXiv preprint arXiv:1611.07400*.

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE communications surveys & tutorials, 16*(3), 1617-1634.

Prasath, M. K., & Perumal, B. (2019). A meta- heuristic Bayesian network classification for intrusion detection. *International Journal of Network Management, 29*(3), e2047.

Ren, K., Yuan, S., Zhang, C., Shi, Y., & Huang, Z. (2023). CANET: A hierarchical CNN-Attention model for Network Intrusion Detection. *Computer Communications, 205*, 170-181. doi:https://doi.org/10.1016/j.comcom.2023.04.018

Revathi, S., & Malathi, A. (2013). A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT), 2*(12), 1848-1853.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security, 86*, 147-167.

Sadiku, M. N. (2019). *Emerging internet-based technologies*: CRC Press.

Said, R. B., Sabir, Z., & Askerzade, I. (2023). CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking With Hybrid Feature Selection. *IEEE Access, 11*, 138732-138747. doi:10.1109/ACCESS.2023.3340142

Sathya, R., & Thangarajan, R. (2015). *Efficient anomaly detection and mitigation in software defined networking environment.* Paper presented at the 2015 2nd international conference on electronics and communication systems (ICECS).

Scott-Hayward, S., O'Callaghan, G., & Sezer, S. (2013). *SDN security: A survey.* Paper presented at the 2013 IEEE SDN For Future Networks and Services (SDN4FNS).

Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., . . . Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine, 51*(7), 36-43.

Shah, A. A., Khan, Y. D., & Ashraf, M. A. (2020). Attacks Analysis of TCP and UDP of UNCW-NB15 Dataset.

Sharma, N., & Mukherjee, S. (2012). A novel multi-classifier layered approach to improve minority attack detection in IDS. *Procedia Technology, 6*, 913-921.

Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013). *Avant-guard: Scalable and vigilant switch flow management in software-defined networks.* Paper presented at the Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.

Shin, S. W., Porras, P., Yegneswara, V., Fong, M., Gu, G., & Tyson, M. (2013). *Fresco: Modular composable security services for software-defined networks.* Paper presented at the 20th annual network & distributed system security symposium.

Shovic, J. C., & Simpson, A. (2019). *Python All-in-one for Dummies*: John Wiley & Sons.

Song, C., Park, Y., Golani, K., Kim, Y., Bhatt, K., & Goswami, K. (2017). *Machine-learning based threat-aware system in software defined networks.* Paper presented at the 2017 26th international conference on computer communication and networks (ICCCN).

Stallings, W. (2015). *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*: Addison-Wesley Professional.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). *LSTM neural networks for language modeling.* Paper presented at the Thirteenth annual conference of the international speech communication association.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). *Deep learning approach for network intrusion detection in software defined networking.* Paper presented at the 2016 international conference on wireless networks and mobile communications (WINCOM).

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2018). *Deep recurrent neural network for intrusion detection in sdn-based networks.* Paper presented at the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft).

Taylor, R., Baron, D., & Schmidt, D. (2015, 21-23 Oct. 2015). *The world in 2025 - predictions for the next ten years.* Paper presented at the 2015 10th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT).

Underdahl, B., & Kinghorn, G. (2015). Software Defined Networking for Dummies. *Cisco Special Edition, John Wiley & Sons, Inc, Hoboken, New Jersey*.

Vasudevan, A. B., Dai, D., & Van Gool, L. (2021). Talk2Nav: Long-Range Vision-and-Language Navigation with Dual Attention and Spatial Memory. *International Journal of Computer Vision, 129*(1), 246-266. doi:10.1007/s11263-020-01374-3

Wang, J., Wang, Y., Hu, H., Sun, Q., Shi, H., & Zeng, L. (2013). *Towards a security-enhanced firewall application for openflow networks.* Paper presented at the Cyberspace Safety and Security: 5th International Symposium, CSS 2013, Zhangjiajie, China, November 13-15, 2013, Proceedings 5.

Wang, T., Zhao, Y., Zhu, L., Liu, G., Ma, Z., & Zheng, J. (2020, 13-15 Nov. 2020). *Lung CT image aided detection COVID-19 based on Alexnet network.* Paper presented at the 2020 5th International Conference on Communication, Image and Signal Processing (CCISP).

Wen, X., Chen, Y., Hu, C., Shi, C., & Wang, Y. (2013). *Towards a secure controller platform for openflow applications.* Paper presented at the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking.

Wencheng, C., Xiaopeng, G., Hong, S., & Limin, Z. (2018). *Offline Chinese signature verification based on AlexNet.* Paper presented at the Advanced Hybrid Information Processing: First International Conference, ADHIP 2017, Harbin, China, July 17–18, 2017, Proceedings 1.

Yang, H., Bai, Y., Chen, T., Shi, Y., Yang, R., & Ma, H. (2022). *Intrusion detection model for power information network based on multi-layer attention mechanism.* Paper presented at the 2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC).

Zhou, Y., & Wang, M. (2020, 2020//). *Remote Sensing Image Classification Based on AlexNet Network Model.* Paper presented at the Frontier Computing, Singapore.