



DDoS attack detection and mitigation using deep neural network in SDN environment

Vanlaluata Hnamte^{a,*}, Ashfaq Ahmad Najar^b, Hong Nhun-Nguyen^c, Jamal Hussain^a, Manohar Naik Sugali^b

^a Department of Mathematics and Computer Science, Mizoram University, Tathril, Aizawl, 796004, Mizoram, India

^b Department of Computer Science, Central University of Kerala, Tejaswini Hills, Periy, 671320, Kerala, India

^c Department of Information Technology, Viet Tri University of Industry, Tien Son Street, Viet Tri City, 29000, Phu Tho Province, Viet Nam



ARTICLE INFO

Keywords:

Deep learning
Deep neural network
SDN
DDoS detection
Distributed denial of service attack
Anomaly detection

ABSTRACT

In the contemporary digital landscape, the escalating threat landscape of cyber attacks, particularly distributed denial-of-service (DDoS) attacks, has become a paramount concern for network security. This research introduces an innovative approach to DDoS detection leveraging a deep neural network (DNN) architecture rooted in deep learning (DL) principles. The proposed model exhibits a scalable and adaptable framework, enabling meticulous analysis of network traffic data to discern intricate patterns indicative of DDoS attacks. To validate the efficacy of our methodology, rigorous evaluations were conducted using authentic real-world traffic data. The results unequivocally establish the superiority of our DNN-based approach over traditional DDoS detection techniques. This research holds significant promise for bolstering network security, particularly within the dynamic landscape of software-defined network (SDN) environments. The study's findings contribute to the continual refinement and eventual deployment of advanced measures in fortifying digital infrastructure against the evolving threat landscape. Performance metrics, including detection accuracy and loss rates, further emphasize the effectiveness of our approach across different datasets. With detection accuracy rates of 99.98%, 100%, and 99.99% for the InSDN, CICIDS2018, and Kaggle DDoS datasets, respectively, coupled with low loss rates, our DNN-based model demonstrates robust capabilities in mitigating contemporary DDoS threats. This study not only presents a novel DDoS detection approach within SDN infrastructures but also offers insights into practical implications and challenges associated with deploying DNNs in real-world SDN environments. Network security professionals can benefit from the nuanced perspectives provided, contributing to the ongoing discourse on fortifying digital networks against evolving cyber threats.

1. Introduction

In an era characterized by an ever-increasing reliance on technology, the significance of cybersecurity has soared to unprecedented heights. With the pervasive adoption of digital communication, the proliferation of the Internet of Things (IoT), the ubiquity of cloud computing, and the omnipresence of mobile devices, the attack surface for cyber threats has expanded exponentially. Cybersecurity, encompassing a multifaceted spectrum of practices, is dedicated to the safeguarding of computer systems and networks against unauthorized access, theft, damage, or disruption of services. The imperative for cybersecurity has escalated in tandem with the evolution of cyber threats, which have grown progres-

sively sophisticated. The ramifications of cyberattacks are profound, encompassing substantial financial losses, severe damage to reputation, and the perilous compromise of sensitive data. These malevolent incursions are indiscriminate, targeting individuals, organizations, and even governments, thereby bestowing upon them far-reaching and potentially devastating consequences.

Among the myriad cyber threats that loom large on the contemporary cybersecurity landscape, DDoS attacks emerge as a formidable adversary, wielding the potential to wreak havoc upon online services and enterprises alike. DDoS attacks, characterized by their modus operandi of inundating a network or server with a deluge of traffic, achieve the nefarious objective of rendering the target inaccessible

* Corresponding author.

E-mail addresses: vanlaluata.hnamte@gmail.com (V. Hnamte), ishfaqnajar@gmail.com (A.A. Najar), nhungnguyen.uet@gmail.com (H. Nhun-Nguyen), jamal.mzu@gmail.com (J. Hussain), manoharamen@cukerala.ac.in (M.N. Sugali).

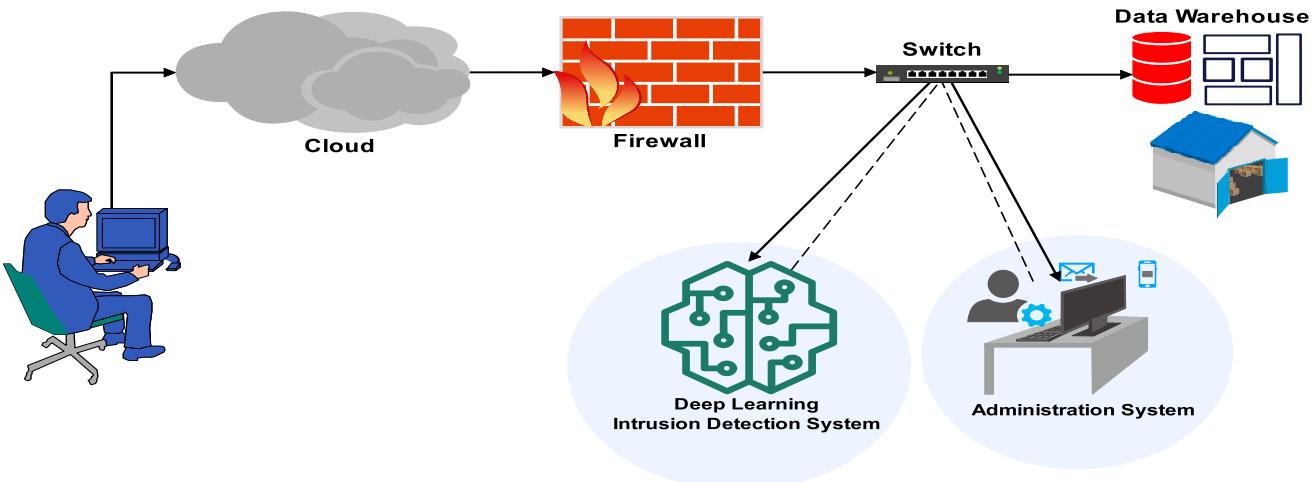


Fig. 1. Mapping vulnerabilities within the architecture of SDN at a conceptual level.

to legitimate users. This offensive prowess is often amplified by the malefactors harnessing a network of compromised computers, colloquially termed a botnet, thereby obfuscating the identification of the attacker's origin. DDoS attacks, notorious for their deleterious effects, precipitate substantial financial losses, inflict reputational harm and orchestrate debilitating disruptions to online services. These pernicious attacks have cast a wide net, ensnaring entities ranging from modest small businesses to sprawling corporate enterprises, oftentimes serving as a conduit for cybercriminals to extort monetary gains or clandestinely infiltrate repositories of sensitive data.

However, in the face of this escalating menace, traditional methodologies for the detection of DDoS attacks have proven increasingly ineffectual in stemming the torrential tide of these network-borne attacks. The inexorable march of technology begets an ever-expanding landscape fraught with novel complexities and nuances, demanding correspondingly sophisticated countermeasures. Conventional detection techniques, constrained by their capacity to encapsulate the intricate relationships and patterns latent within data, falter in their ability to furnish optimal intrusion detection outcomes, rendering them susceptible to the perils of both false positives and false negatives.

In this swiftly evolving cybersecurity milieu, SDN has emerged as a salient technological bulwark, poised to bolster network security through its centralized vantage point for traffic surveillance and dynamic resource management. Concurrently, DL, a specialized domain within the purview of Artificial Intelligence, has surfaced as a potent instrument for detecting DDoS attacks via its aptitude for dissecting network traffic patterns. The fusion of SDN and DL holds promise in heralding a paradigm shift in the realm of DDoS detection, a testament to its burgeoning adoption and investigation within recent academic research endeavors. Fig. 1 illustrates the conceptual framework for mapping vulnerabilities within the SDN architecture. It visually represents the possibility of identification and analysis using DL, providing a comprehensive overview of security considerations in an SDN environment.

SDN, through its radical partitioning of the control plane and data plane within network devices, empowers centralized governance and orchestration of network resources. This newfound vantage furnishes network administrators with augmented visibility and precise control over network traffic, thereby rendering it an ideal bastion for the detection and amelioration of DDoS attacks. Traditional DDoS detection modalities, grounded in the tenets of signature-based detection and anomaly-based detection, grapple with limitations in the contemporary milieu.

The efficacious mitigation of DDoS attack vectors hinges upon the formulation of robust detection and mitigation strategies. These encompass a multifaceted arsenal comprising Intrusion Detection Systems

(IDS), firewalls, and traffic filtration mechanisms. In concert with these proactive measures, the crystallization of a well-conceived incident response blueprint is imperative, replete with protocols for alleviating the impact of an attack and expedited service restoration. The urgency of devising a more precise and efficacious modality for the detection of nascent attack modalities is underscored within the purview of cloud computing networks, SDN, and servers.

An expedited and astute identification of malevolent traffic streams offers the potential for real-time countermeasures against DDoS onslaughts. Were a prospective target equipped with the capability for instantaneous detection, the resultant impact of such an attack might be diminished or potentially negated. Furthermore, the assimilation of a detection system into the fabric of the broader Internet, affording networks the acumen to recognize egregiously aggressive traffic and interdict its propagation, could substantially curtail the quantum of traffic amenable to generation during an attack. A bespoke machine learning model, meticulously trained on conventional network data and fine-tuned for expeditious and precise detection, stands as a viable conduit for achieving both these objectives.

DL, as a subdomain of machine learning, boasts a panoply of anomaly detection models that can be categorized into three principal archetypes: supervised, unsupervised, and semi-supervised. Supervised learning models, replete with their predilection for data labeled with ground-truth annotations, exhibit an elevated capacity for predictive modeling and classification tasks. However, the relative scarcity of labeled training data attenuates the practical applicability of supervised models, thereby motivating the exploration of alternative paradigms.

This paper embarks on an exploration of the efficacy of DL techniques in the domain of DDoS attack detection within SDN environments. We proffer a comprehensive framework that harnesses DNN to scrutinize network traffic data, thereby discerning latent patterns indicative of incipient DDoS attacks. The inherent scalability and adaptability of our methodology, designed to accommodate the detection of nascent attack modalities as they surface, distinguish our approach. Our empirical investigations, underpinned by real-world traffic data, substantiate the superiority of our method vis-a-vis traditional DDoS detection modalities. The outcomes of our study corroborate the potential of DL in fortifying network security within SDN domains, hinting at the prospect of further refinement and subsequent deployment within practical network infrastructures. Our approach exhibits marked advantages over traditional techniques in terms of accuracy, detection rate, and false-positive rate.

The subsequent sections of this paper are structured as follows: Section 2 furnishes an expository backdrop to this research and proffers an overview of related work. Section 3 expounds upon the conceptual

framework underpinning our proposed models, detailing their design and encapsulating the intricacies of their implementation and empirical testing. Section 4 delineates the empirical findings emanating from our testing endeavors, juxtaposing them with antecedent research outcomes. Finally, Section 5 furnishes a succinct denouement to the paper and charts a course for prospective research trajectories.

2. Recent studies

In the last decade, substantial research efforts have been dedicated to integrating DL techniques into network intrusion detection, particularly focusing on feature selection as a crucial precursor. Recent studies have exhibited a notable concentration on employing DNNs for detecting DDoS attacks within the dynamic landscape of SDN environments (Yan et al., 2016; Karan et al., 2018; Ali et al., 2020).

Zainudin et al. (2022) introduced a framework that harnesses the power of DNNs to analyze network traffic data and identify DDoS attacks. Leveraging the benefits of SDN, this framework provides a centralized perspective of network traffic, facilitating dynamic control of network resources to effectively mitigate the impact of DDoS attacks. The authors rigorously evaluated their approach using real-world traffic data, demonstrating its superiority over traditional DDoS detection techniques. However, it's imperative to highlight that this framework was not assessed on an IoT-specific dataset, potentially constraining its effectiveness in detecting DDoS attacks targeted specifically at IoT devices. Additionally, the study omitted the evaluation of certain attack types and samples, which might influence the comprehensiveness of the proposed approach in detecting a wide spectrum of DDoS attacks.

Santos-Neto et al. (2022) proposed a hybrid approach for DDoS attack detection in SDN environments, combining unsupervised and supervised machine learning techniques. Their method deploys clustering algorithms to identify anomalous network traffic patterns, followed by DNN classification to distinguish between DDoS attacks and benign traffic. The authors conducted thorough evaluations on a dataset of network traffic data, revealing high accuracy in DDoS attack detection while minimizing false alarms. However, it's essential to recognize that evaluating the approach solely on a network traffic dataset might not fully encapsulate the diversity of real-world DDoS attack scenarios, potentially limiting the generalizability of the results.

Hussain and Hnamte (2021a) proposed a DL-based approach employing feed-forward multi-layer perceptrons (MLP) for intrusion detection, achieving an impressive accuracy rate of approximately 98%. This underscores the effectiveness of DNNs in the field of IDS, where precise identification of malicious activities is paramount for maintaining network security. It's noteworthy that the experiment primarily focused on traditional network environments and did not specifically assess the model's performance with newer datasets tailored for SDN.

Elmasry et al. (2020) employed a feature selection technique in conjunction with several models, with the Deep Belief Network (DBN) emerging as the top performer. However, a limitation of their study lies in the absence of extensive evaluation of the DBN model across diverse datasets or explicit exploration of its performance in detecting DDoS attacks. Although DBN exhibits comparability to MLP, it follows a different architectural paradigm and tends to excel with more hidden layers. However, the increased complexity introduced by additional hidden layers renders DBN models generally slower than MLP models with equivalent layer configurations. Nevertheless, their model achieved an accuracy exceeding 99%. Furthermore, the results showcased a substantial improvement of 4 to 6% in network intrusion detection when compared to models without pretraining, along with a reduction of 5-1% in false alarm rates on similar datasets.

Tang et al. (2016) introduced a DNN model for network intrusion detection in SDN. Their model, trained on a subset of six features se-

lected from the NSL-KDD dataset,¹ achieved a detection rate of 76%. Notably, the authors applied Principal Component Analysis (PCA) to transform NSL-KDD features, followed by feature subset optimization using Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) techniques. These enhanced features were integrated into a Modular Neural Network (MNN) model, resulting in GA achieving a detection rate of 98.2% with a false alarm rate of 1.8%, while PSO achieved 99.4% detection with a false alarm rate of 0.6%. However, it's essential to acknowledge that the evaluation dataset may not entirely capture SDN network characteristics.

Isa and Mhamdi (2020) developed a hybrid DL approach, combining auto-encoding with the Random Forest (RF) algorithm, to combat DDoS attacks in SDNs. Focusing on the native SDN environment, their method aimed for high detection accuracy while minimizing computational overhead. Though their study reported promising results in terms of accuracy and efficiency using realistic datasets, it did not specifically consider the impact of the model on SDN controller performance. Similar to Tang et al. (2016), their approach achieved an accuracy of 98.4% on the NSL-KDD benchmark dataset.

Chanu et al. (2023) proposed a voting-based hybrid feature selection technique for detecting DDoS attacks. They highlighted the limitations of naive feature selection methods, emphasizing the challenge of accurately detecting DDoS attacks. Their hybrid feature selection aimed to reduce dimensions, eliminate redundancy, and identify relevant features, resulting in an impressive accuracy of 98.8% with a low false positive rate of 0.6% and early detection capability.

Li et al. (2018) utilized a bidirectional Recurrent Neural Network (RNN) across SDN layers to detect and block DDoS attacks in real-time. While effective, this approach may face limitations in larger networks with multiple controllers, as RNNs can disrupt controller synchronization, potentially impacting network performance.

Bhuyan et al. (2015) proposed a technique for identifying low- and high-rate DDoS attacks based on correlation coefficients. While the method demonstrated strong correlations between instances of malicious traffic, its effectiveness in detecting single instances of malicious traffic remained unclear.

Pérez-Díaz et al. (2020) proposed an adaptable architectural framework employing machine learning for the detection and mitigation of slow-rate Denial of Service (DoS) attacks. Their modular system exhibited a commendable detection accuracy of 95%, a noteworthy achievement given the inherent difficulty in identifying low-rate DDoS attacks.

Banitalebi Dehkordi et al. (2021) introduced a method for detecting DDoS attacks within SDN environments, boasting an impressive maximum accuracy rate of 99.96%. However, it is imperative to acknowledge the study's reliance on outdated datasets, potentially restricting its capability to recognize emerging DDoS attack patterns.

Berman et al. (2019) conducted an exhaustive inquiry into the realm of DL techniques within the domain of cybersecurity. Their research delved into diverse attack scenarios, underscoring DL's potential in uncovering novel malware variants and zero-day attacks. Moreover, they emphasized the pivotal requirement for standardized benchmark datasets and addressed pertinent considerations related to adversaries when deploying DL in the cybersecurity landscape.

Najar and Manohar Naik (2022) conducted a comprehensive exploration into the realm of machine learning techniques for the detection and classification of DDoS attack packets. They investigated the performance of various algorithms, including Random Forest (RF), multi-layer perceptrons (MLP), Support Vector Machine (SVM), and K-Nearest Neighbor, using the NSL-KDD dataset. Notably, RF exhibited exceptional accuracy, achieving 99.13% on both training and validation data and 97% on the full test dataset. Meanwhile, MLP showcased impressive accuracies of 97.96% on training data, 98.53% on validation data, and 74% on the full test dataset. These results underscore the consider-

¹ <https://www.unb.ca/cic/datasets/nsl.html>.

able potential of machine learning techniques for the accurate detection and classification of DDoS attacks.

Cil et al. (2021) demonstrated the effectiveness of DNN models in DDoS attack detection, leveraging the CICDDoS2019 dataset. Their DNN model exhibited exceptional performance metrics, including an F1-Score of 0.9998, Accuracy of 0.9997, Precision of 0.9999, and Recall of 0.9998 for Dataset 1. However, for Dataset 2, performance exhibited a slight decline, with an F1-Score of 0.8721, Accuracy of 0.9457, Precision of 0.8049, and Recall of 0.9515. These findings highlight the necessity for further research to enhance multiclass classification, particularly for Dataset 2.

Ferrag et al. (2020) conducted an exhaustive analysis of DL techniques in the realm of intrusion detection, categorizing diverse cybersecurity datasets. Their study shed light on the limitations of Convolutional Neural Networks (CNNs) in capturing long-time-dependent features of IoT traffic. Consequently, they emphasized the critical need for future research efforts in this domain.

Aljabri et al. (2021) introduced DeepDefense, a DDoS attack detection method rooted in Recurrent Neural Networks (RNNs). DeepDefense effectively incorporates fully connected layers, RNNs, and CNNs to bolster detection accuracy while significantly reducing error rates. While this approach shows great promise, the study recommended further exploration of hybrid models and the addressing of challenges related to the detection of malicious URLs and traffic in encrypted networks and IoT environments.

Wei et al. (2021) addressed the limitations of traditional machine learning techniques in DDoS attack detection by introducing a hybrid DL approach. Their model combines an autoencoder for feature extraction with a multi-layer perceptron for precise attack classification. Experimental results on the CICDDoS2019 dataset showcased remarkable performance, with an accuracy rate exceeding 98%. However, it is essential to conduct further research to assess the model's adaptability to diverse network environments and various attack variations.

Yuan et al. (2017) presented an effective approach for detecting DDoS attacks employing a DL model that incorporates improved feature selection. Their method incorporates feature selection based on the chi-square test and combines a bidirectional long short-term memory (Bi-LSTM) with a CNN for attack detection. Although their results revealed significant reductions in error rates, additional testing of the model in real-time scenarios is warranted to evaluate its performance comprehensively.

Said Elsayed et al. (2020) elevated anomaly detection by amalgamating an LSTM-autoencoder with the one-class SVM (OC-SVM) algorithm. This combined approach, termed LSTM-Autoencoder-OC-SVM, exhibited enhanced accuracy compared to OC-SVM in isolation. Remarkably, LSTM-Autoencoder-OC-SVM achieved an accuracy of 90.5%, surpassing the OC-SVM's accuracy of 87.5.

Fouladi et al. (2022) introduced an innovative technique for the detection and mitigation of DDoS attacks within SDN environments. Their approach combines discrete wavelet transforms with autoencoder neural networks to augment the accuracy of IDSs. By leveraging the discrete wavelet transform to extract statistical features from network traffic data and employing an autoencoder neural network for unsupervised learning, their method excels in the precise identification of DDoS attack patterns.

Hnamte and Hussain (2023a) proposed a novel approach for detecting DDoS attacks using hybrid DL-based DNNs. Their models, trained on the CIC-IDS2017 and CIC-DDoS2019 datasets, achieved a remarkable accuracy of 99.9% with an impressively low loss rate of 0.0025. While these results highlight the effectiveness of DNNs in identifying and classifying DDoS attacks, further research is needed to evaluate their applicability in SDN environments.

Swami et al. (2023) employed OpenFlow Random Host Mutation (OF-RHM) to enhance system security against attacks, reducing the likelihood of successful attacks by periodically changing host IP addresses randomly. However, this study did not focus on DL-based systems.

Agarwal et al. (2021) presented a novel approach combining feature selection with a whale optimization algorithm and a DNN to effectively resist DDoS attacks. Their model also homomorphically encrypted and securely stored data in the cloud to enhance security. Simulation results demonstrated an impressive accuracy of 95.35% in detecting DDoS attacks. A swarm intelligence technique was utilized to identify optimal features, and the Aquila optimizer assigned desirable weights to selected features after the feature selection process.

Various techniques are available for visualizing different types of data. For instance, Sayed et al. (2022) utilized t-Distributed Stochastic Neighbor Embedding (t-SNE) to represent the data arrangement of the InSDN dataset. t-SNE is a method that transforms high-dimensional data into a lower-dimensional space while preserving the neighborhood relationships among the data points. The results indicate that normal and attack samples exhibit similar characteristics. As a result, it becomes challenging to achieve linear separation between these two classes, highlighting the complexity of intrusion detection issues in network traffic.

In this study (Amaizu et al., 2021), the authors have made notable advancements in the realm of DDoS detection within 5G/B5G networks by introducing a novel approach. They leverage hybrid DNN models, enriched with Pearson Correlation Coefficient (PCC) feature extraction, as a robust methodology to enhance the accuracy and efficiency of DDoS detection. The experimental results showcase the efficacy of this approach, with an impressive accuracy rate of 99.66% and a minimal loss of 0.011, as verified on the CICDDoS2019 dataset. Significantly, the hybrid model outperforms all other models, except for a CNN ensemble, underscoring the superiority of the proposed hybrid approach. Nevertheless, the study judiciously addresses a pivotal concern regarding the inherent complexity of the proposed hybrid DNN models. This complexity introduces valid apprehensions about potential delays in detection times, posing a potential challenge to the real-time applicability of the framework, particularly in dynamic and high-traffic environments.

In this study (Mishra et al., 2023), the authors proposed a framework for the detection and classification of DDoS attacks. The study utilizes the CICDDoS2019 dataset as the basis for its investigation into DDoS attack categorization and prediction. The primary methodology involves the application of the Extra Tree Classifier, a machine learning algorithm used for feature optimization. The framework achieves notable success by incorporating the AdaBoost Classifier, resulting in an impressive 99.87% accuracy. The Extra Tree Classifier is specifically employed to distill 25 key attributes crucial for the effective categorization and prediction of DDoS attacks.

In this study (Chouhan et al., 2023), the effectiveness of various machine learning classifiers, including Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbors (K-NN), Extreme Gradient Boosting, and Naive Bayes (NB), was rigorously assessed using a generated dataset for real-time intrusion detection within SDN environments. The results indicate that SVM outperformed other classifiers with remarkable metrics, achieving 99.398% accuracy, 99.413% precision, 99.397% recall, 0.718% False Acceptance Rate (FAR), 0.995 Area Under the Curve (AUC), and 99.400% F1 value. While the study demonstrates the high performance of SVM in real-time intrusion detection, it acknowledges certain limitations. The real-time performance of the proposed model in real-world environments may be impacted by longer detection times due to the complexity of the model structure. Additionally, the relatively small size of the dataset raises concerns about the generalizability of the findings.

While these studies from Table 1 demonstrate the potential of DL in network intrusion detection, their applicability to SDN environments varies. Some studies explicitly target SDN, showcasing adaptability and effectiveness, while others may require further evaluation in SDN contexts. Researchers should consider the nuances of SDN architectures and traffic patterns when applying DL techniques, ensuring their relevance and efficiency in these dynamic environments.

Table 1

Summary of existing literature.

Reference	Year	Model	Dataset	Accuracy	Loss	Recall	Precision	F1-Measure
Li et al. (2018)	2018	DNN	ISCX	98%	*	*	99%	*
Elmasry et al. (2020)	2020	Double PSO + DBN	CICIDS2017, NSL-KDD	99.91% 99.79%	*	99.92% 99.81%	99.99% 99.83%	99.95% 99.82%
Isa and Mhamdi (2020)	2020	Hybrid Approach	NSL-KDD	98.4%	*	*	*	*
Pérez-Díaz et al. (2020)	2020	MLP	CICDDoS2017	95%	*	94.51%	95.46%	94.98%
Banitalebi Dehkordi et al. (2021)	2020	RandomTree, REPTree	UNB-ISCX, CTU-13, and ISOT	99.96% 98.55%	*	99.44% 98.10%	97.15% 99.64%	98.28% 98.86%
Ferrag et al. (2020)	2020	DNN RNN CNN DNN RNN CNN	CICIDS2018 BoT-IoT	97.28% 97.31% 97.38% 98.22% 98.31% 98.37%	*	*	*	*
Said Elsayed et al. (2020)	2020	LSTM Based AE	InSDN	90.5%	*	93%	93%	93%
Hussain and Hnamte (2021a)	2021	DNN	KDDCUP99, NSL-KDD	98.99% 96.00%	*	98.99% 96.0.%	98.75% 94.12%	98.79% 95.75%
Gil et al. (2021)	2021	DNN	CICDDoS2019	99.9% 94.57%	*	99.98% 95.15%	99.99% 80.49%	99.98% 87.21%
Hussain and Hnamte (2021b)	2021	DNN	KDDCUP99, NSL-KDD	99.69% 97.26%	0.0207 0.1615	99.61% 98.12%	99.37% 97.73%	99.48% 97.83%
Hussain and Hnamte (2021c)	2021	DNN	KDDCUP99, NSL-KDD UNSW-NB15	99.69% 95.38% 81.70%	0.0207 0.1615 0.5245	99.61% 98.12% 81.70%	99.37% 97.73% 74.79%	99.48% 97.83% 77.35%
Wei et al. (2021)	2021	AE-MLP	CICDDoS2019	98.34%	*	98.48%	97.91%	98.18%
Agarwal et al. (2021)	2021	FS-WOA-DNN	CICIDS2017	95.35%	0.0928	90.71%	*	*
Zainudin et al. (2022)	2022	CNN-LSTM	CICDDoS2019	99.50%	*	*	*	*
Santos-Neto et al. (2022)	2022	Hybrid DNN	Self Generated	99%	*	*	*	*
Najar and Manohar Naik (2022)	2022	RF MLP	NSL KDD	97% 74%	0.0656	61.75% 72.93%	90.72% 85.57%	96.61% 77.43%
Fouladi et al. (2022)	2022	AE -NN	MAWI	98.85%	*	*	*	*
Fatani et al. (2022)	2022	CNN	CICIDS2017, NSL-KDD, BoT-IoT, and KDDCUP99	99.91%	*	99.88%	99.88%	99.88%
Hnamte et al. (2023)	2023	LSTM-AE	CICIDS2017, CSE-CICDIS2018	99.99% 99.10%	0.0005 0.0040	99.99% 99.10%	99.99% 99.07%	99.99% 99.02%
Chanu et al. (2023)	2023	Hybrid Approach(MLP-GA)	CICDDoS2017	98.8%	*	*	*	*
Hnamte and Hussain (2023b)	2023	DCNNBiLSTM	CICIDS2018, Edge_IoT datasets	100% 99.64%	0.0000 0.0080	*	*	*

* Not available.

3. Methodology

This section is primarily comprised of three subsections: dataset preparation, model construction, and evaluation of the model. The objective of data preparation is to decrease the time required by the model and maintain the objectivity of the evaluation. The model generation section produces a DNN model and improves its performance through continuous execution until the expected performance is found and meets the value specified. Fig. 2 depicts our proposed method.

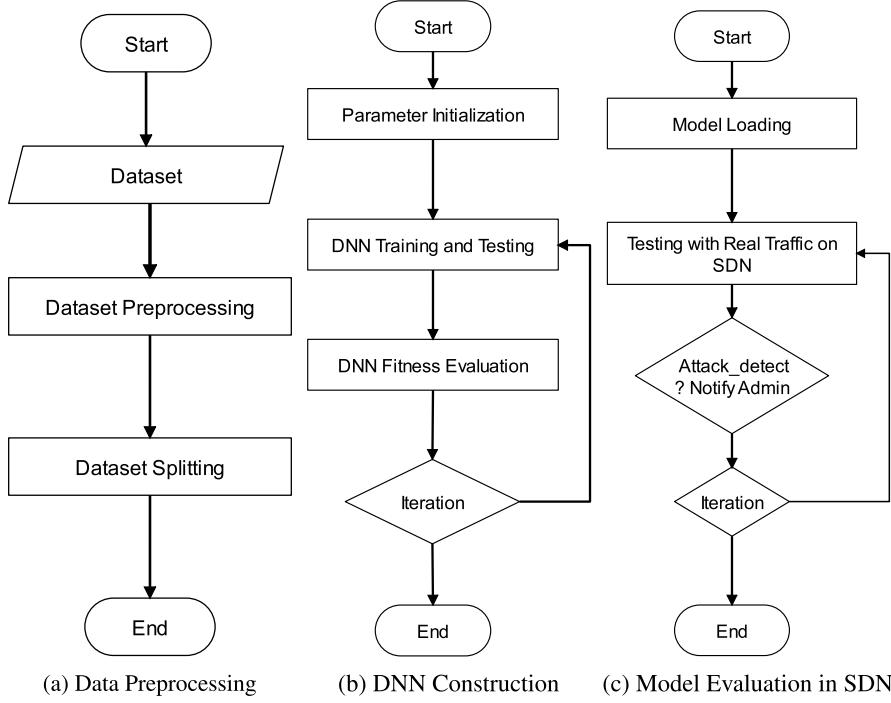
The first flowchart (2a) depicts the process of dataset preparation; the second flowchart (2b) depicts the process of model generation; and the third flowchart (2c) depicts the process of evaluating the generated model in an SDN environment.

3.1. Data preprocessing

Data preprocessing is applied to clean the data, normalize the data, and filter a subset of features. This step is critical since the noise con-

tained in the data might hamper performance. Data cleansing is essential since actual acquired data might contain numerous irrelevant features. We prepared the dataset by removing some features and normalizing the data. The majority of machine learning models can only deal with numerical numbers for training and testing, which is their limitation. Thus, it is important to do a data numericalization to transform any non-numerical values into numerical ones. Basically, there are two approaches to numericalizing data. The first is known as “one-hot encoding,” which assigns a unique binary vector to each kind of nominal characteristic. For example, the InSDN dataset contains nominal characteristics such as protocol type, service, etc.

Algorithm 1 takes as input the dataset D and the desired percentage p for the training set. It shuffles the dataset randomly to ensure a random distribution of samples. It then calculates the number of samples to include in the training set based on the percentage p . The algorithm extracts the first n_{train} samples as the training set D_{train} and the remaining samples as the testing set D_{test} . The algorithm allows for additional data preprocessing steps to be performed on the training and testing sets

**Fig. 2.** Flowchart of the method.**Algorithm 1** Dataset preprocessing and splitting.

Require: Dataset D , Training set percentage p
Ensure: Training set D_{train} , Testing set D_{test}

- 1: Shuffle D randomly
- 2: Calculate the number of samples in the training set: $n_{train} = \text{round}(p \times |D|)$
- 3: Extract the first n_{train} samples from D as D_{train}
- 4: Extract the remaining samples from D as D_{test}
- 5: **Preprocessing:** Perform any required data preprocessing steps on D_{train} and D_{test} , such as feature scaling, normalization, or missing data handling.
- 6: Store D_{train}, D_{test}

if needed, such as feature scaling, normalization, or handling missing data. Optionally, the training set D_{train} can be further split into training and validation sets for tasks like hyperparameter tuning or model selection. Finally, the algorithm stores the training set D_{train} and the testing set D_{test} as the output.

3.2. DNN model

A DNN is a type of artificial neural network (ANN) characterized by a deep architecture with multiple layers, particularly multiple hidden layers between the input and output layers. Each layer in a DNN comprises interconnected nodes, or neurons, and these networks are capable of learning intricate representations of data through a process called DL. DNNs are a fundamental component of DL, a subset of machine learning that focuses on using neural networks with many layers (hence “deep”) to model and solve complex problems.

In the realm of performance, DL models surpass traditional machine learning algorithms, albeit with an associated increase in training time due to their inherent complexity. However, their efficacy is intricately tied to the hyperparameter values employed. Thus, a critical concern when utilizing DL models revolves around the judicious tuning of these hyperparameters, which are pivotal settings dictating the behavior, architecture, and performance of the model for a given task. The challenge lies in the task-specific variability of these hyperparameter values, necessitating a thoughtful selection process before initiating the learning process.

The Algorithm 2 outlines the proposed DNN model, a critical phase where the network learns to map input data to desired output by ad-

justing its weights and biases. Below is a detailed breakdown of each step:

1. **Input Parameters:**
 - \mathbf{x} : Input data vector, where each element represents a feature.
 - \mathbf{y} : Output data vector, corresponding to the desired output.
 - L : Number of hidden layers in the DNN.
 - N : Number of neurons per hidden layer.
 - α : Learning rate, determining the step size during weight updates.
 - σ : Activation function, introducing non-linearity to the model.
2. **Initialization:** Initialize weights and biases for all layers randomly. This step involves assigning initial values to the parameters that the model will learn during training.
3. **Training Loop:** The training process is iterative and occurs within a repeat-until loop, which continues until convergence or the maximum number of iterations is reached.
4. **Iteration Over Training Data:** For each iteration over the training data (N_{train} instances):
 - **Feedforward:** Calculate the predicted output $\hat{\mathbf{y}}$ by passing the input \mathbf{x}_i through the DNN using feedforward propagation.
 - **Compute Loss:** Calculate the loss E using a specified loss function, commonly mean squared error $(\mathbf{y}_i - \hat{\mathbf{y}})^2$.
 - **Backpropagation:** Compute the gradients ∇E of the loss with respect to all weights and biases using backpropagation. This step involves calculating how much the loss would change with respect to each parameter.
 - **Update Weights and Biases:** Update the weights and biases using the gradients and the learning rate through the process of gradient descent: $\theta_{t+1} = \theta_t - \alpha \nabla E$.
5. **Convergence Check:** The algorithm repeats the training loop until convergence, where the model’s performance stabilizes, or the maximum number of iterations is reached.

The input data \mathbf{x} and output data \mathbf{y} are first given as input to the algorithm, along with the number of hidden layers, number of neurons per hidden layer, learning rate, and activation function. The weights and biases for all layers are then initialized randomly.

Table 2

Hyperparameters and their domains.

Hyperparameter	Domain	Type
Learning rate	0.0001	Continuous
Dropout rate	0.1	Continuous
Number of hidden layers	4	Step = 1
Numbers of neurons of hidden layers	128, 256, 128	Step = 1
Number of epochs	30	Step = 1
Batch size	128	Step = 1
Optimizer	Adam	Step = 1
Layer type	Dropout, Dense	Step = 1
Activation function	Softmax, Relu	Step = 1

The algorithm then performs a repeat-until loop for a maximum number of iterations or until convergence is achieved. For each iteration, the algorithm processes the training data by first feeding it forward through the network to calculate the output \hat{y} . The loss function E is then computed as the mean squared error between the predicted output and the actual output.

In the next step, the gradients of the loss function with respect to all the weights and biases are computed using backpropagation. These gradients are then used to update the weights and biases of the network through the process of gradient descent. The learning rate α determines the step size of the weight and bias updates.

The process is repeated for all training data until convergence is achieved or the maximum number of iterations is reached. The trained DNN model with the updated weights and biases is then returned as the output of the algorithm.

Algorithm 2 DNN algorithm.

```

Require: Input data  $x = x_1, x_2, \dots, x_n$ , output data  $y = y_1, y_2, \dots, y_m$ , number of hidden layers  $L$ , number of neurons per hidden layer  $N$ , learning rate  $\alpha$ , and activation function  $\sigma$ 
Ensure: Trained DNN model
1: Initialize weights and biases for all layers randomly
2: repeat
3:   for  $i \leftarrow 1$  to  $N_{train}$  do
4:     Feedforward: Calculate output  $\hat{y}$  of the DNN for input  $x_i$ 
5:     Compute the loss  $E = \frac{1}{2}(y_i - \hat{y})^2$ 
6:     Backpropagation: Compute gradients  $\nabla E$  with respect to all weights and biases
7:     Update weights and biases using gradient descent:  $\theta_t + 1 = \theta_t - \alpha \nabla E$ 
8:   end for
9: until convergence or maximum number of iterations is reached
10: return Trained DNN model with weights and biases

```

The three sequential steps of our proposed technique are preprocessing, model training, and evaluation in an SDN environment. It begins with the preparation phase, during which the primary parameters and a list of desired hyperparameters and their default domains are specified. There were nine hyperparameters defined: learning rate, decay, momentum, number of epochs, batch size, optimizer, initialization function, number of hidden layers, layer type, dropout rate, activation function, and number of hidden layer neurons. The specified hyperparameters and their default domains are shown in Table 2.

Table 2 provides an overview of the proposed DNN hyperparameters and their corresponding settings for a specific model.

- Learning rate:** The learning rate determines the step size at which the model adjusts its parameters during training. A learning rate of 0.0001 indicates a small step size, which helps ensure gradual convergence and prevents overshooting.
- Dropout rate:** Dropout is a regularization technique that randomly sets a fraction of input units to 0 during training to prevent overfitting. A dropout rate of 0.1 means that 10% of the input units will be dropped out or set to 0.
- Number of hidden layers:** This hyperparameter defines the depth of the neural network model. Having more hidden layers allows the

model to learn more complex representations of the input data. In this case, the model has four hidden layers.

- Numbers of neurons in hidden layers:** Each hidden layer contains a certain number of neurons, which determine the capacity and complexity of the model. The given values of 128, 256, and 128 indicate that the first and third hidden layers have 128 neurons, while the second hidden layer has 256 neurons.
- Number of epochs:** An epoch refers to a complete pass through the entire training dataset. The number of epochs determines how many times the model will be trained on the entire dataset. In this case, the model is trained for 30 epochs.
- Batch size:** During training, the dataset is divided into smaller batches, and the model updates its parameters after processing each batch. The batch size of 128 means that the model updates its parameters after processing 128 samples.
- Optimizer:** The optimizer is responsible for updating the model's parameters based on the computed gradients during training. The "Adam" optimizer is commonly used and adapts the learning rate for each parameter individually.
- Layer type:** This hyperparameter specifies the types of layers used in the neural network model. The model includes "Dropout" layers, which help regularize the model by randomly dropping out units, and "Dense" layers, which are fully connected layers.
- Activation function:** Activation functions introduce non-linearities to the model, allowing it to learn complex relationships in the data. The activation functions used in this case are "Softmax" and "Relu". Softmax is often used for multi-class classification, while Relu is a popular choice for hidden layers due to its simplicity and effectiveness in handling non-linearities.

The hyperparameters chosen for our proposed model wield significant influence over its performance. The strategic adjustment of these hyperparameters holds the potential to optimize the model, aligning it more effectively with the training data and enhancing its capacity for generalization when confronted with novel, previously unseen data.

In the context of this study, the hyperparameters detailed in the Table 2 assume pivotal roles in the training process of our DNN model. The meticulous selection of suitable values for each hyperparameter embodies the authors' overarching objective: to harness these hyperparameters' potential to bolster the model's accuracy while concurrently mitigating the loss function. This intricate hyperparameter tuning process aims to calibrate the model to deliver superior performance across various data scenarios, equipping it to make more precise predictions and exhibit enhanced adaptability to uncharted data domains.

3.3. Model loading in SDN

Once the model is trained, we can use it to detect DDoS attacks in SDN by following the steps below.

- Capture Network Traffic:** We need to capture the incoming network traffic in real-time using the SDN controller.
- Data Preprocessing:** Data preprocessing is optional in our study. One can preprocess the captured data by normalizing it and converting it into a format that the DNN model can use as input.
- Run the Model:** We load the saved model, then feed the captured data as input to the trained DNN model to obtain the predicted output. Determine an appropriate threshold for the model's output to classify instances as normal or attacks. This threshold can be based on a predefined value or determined dynamically based on the desired balance between false positives and false negatives.
- Attack Detection:** Compare the output predictions with the threshold to identify whether each instance of network traffic is classified as normal or an attack. Instances exceeding the threshold are considered attacks, while those below the threshold are classified as normal traffic.

- **Alert or Mitigate:** Based on the classification results, appropriate actions can be taken, such as generating alerts, logging events, or implementing mitigation strategies to protect the SDN environment from the detected attacks.

Algorithm 3 DDoS attack detection using DNN model.

Require: Input network traffic data, DNN model weights **W** and biases **B**
Ensure: DDoS attack detection results

- 1: Load the DNN model weights **W** and biases **B**
- 2: Initialize an empty list to store attack instances
- 3: **for** $i \leftarrow 1$ to $N_{samples}$ **do**
- 4: Pass the network traffic data x_i through the DNN model
- 5: Perform forward propagation using the loaded model weights **W** and biases **B**
- 6: Obtain the output prediction \hat{y}_i from the DNN model
- 7: **if** $\hat{y}_i > threshold$ **then**
- 8: Add x_i to the list of detected attack instances
- 9: **end if**
- 10: **end for**
- 11: **return** List of detected attack instances

Algorithm 3 provides a comprehensive depiction of the DDoS detection process using a DNN model. It elucidates the intricate steps involved in the evaluation of input network traffic data for the identification of potential DDoS attacks.

The algorithm commences by loading the essential components of the DNN model, namely its weights (**W**) and biases (**B**). These parameters are instrumental in shaping the model's predictive capabilities. Subsequently, the algorithm takes as input the network traffic data, which serves as the raw material for the detection process.

The process unfolds within a structured loop, iteratively processing each sample within the network traffic data. For every sample, a critical step is executed: the application of forward propagation through the DNN model. This pivotal operation is enacted by leveraging the pre-loaded model weights and biases. The outcome of this operation materializes as an output prediction (\hat{y}_i) originating from the DNN model.

The algorithm operates on a binary decision mechanism, hinging on the comparison of this output prediction against a predefined threshold. If the prediction surpasses this threshold, signifying a substantial likelihood of a DDoS attack, the corresponding instance of network traffic (x_i) is promptly enlisted in the growing roster of detected attack instances.

Ultimately, the algorithm culminates in the aggregation of these detected attack instances into a dedicated list. This list serves as a concise yet comprehensive record of network traffic samples deemed to correspond to DDoS attacks. It encapsulates the primary goal of the algorithm: the precise identification of instances within the network traffic data that exhibit characteristics indicative of DDoS attacks. Upon completion, the algorithm delivers this list of detected attack instances, thereby fulfilling its intended purpose.

In our research endeavor, we have meticulously devised a DNN model tailored specifically for the detection of DDoS attacks within the context of SDN. This model has been systematically stored and meticulously loaded to facilitate the rapid and efficient detection of network attacks, thereby fortifying the security posture of SDN-based networks.

The core of our detection model hinges upon a meticulously crafted process that encompasses the collection of network traffic data, the meticulous design and training of a bespoke DNN architecture, and a comprehensive evaluation of the model's performance. Post-training, our model seamlessly transitions into real-time operation, where it assumes the pivotal role of detecting and mitigating DDoS attacks. This is achieved through the continuous capture of network traffic, which is subsequently fed as input to our expertly trained DNN model. Our pioneering approach to the detection of DDoS attacks is grounded in a profound insight. We surmise that the utilization of the InSDN dataset,

originating from an SDN environment, holds the potential to significantly ameliorate the detection process. This, in turn, can yield a tangible reduction in detection errors and a marked enhancement in detection accuracy. Consequently, the effective detection of DDoS attack scenarios pivots on two critical pillars: the optimization of traffic data and the deployment of an expeditious and highly proficient detection model.

4. Results and discussions

In this section, we embark on a comprehensive exploration of our experimental methodology. We initiate this journey by providing an insightful summary of the datasets meticulously curated for the purpose of our experiments. Subsequently, we delve into the precise details of the evaluation metrics meticulously chosen to gauge the efficacy and performance of our proposed model. Following this, we meticulously outline the intricate facets of the attack scenario that we have meticulously crafted to rigorously assess the capabilities of our proposed model. We also offer a detailed exposition of the specificities of our experimental setup, meticulously describing the network topology, configuration, and all relevant parameters that govern our experimentation.

The results of our comprehensive experimentation endeavor have unveiled the formidable capabilities of DNN models in the realm of DDoS attack detection within SDN environments. Our approach has achieved a remarkable accuracy rate of 99.9%, showcasing its prowess in accurately distinguishing malicious traffic from benign data flows. Furthermore, the false-positive rate, a crucial metric in the context of intrusion detection, stands at a mere 0.003%. These exceptional results underscore the potential and promise of DNN-based approaches in bolstering the security posture of SDN-based networks against the menace of DDoS attacks.

4.1. Dataset

In the realm of cybersecurity research, IDS datasets are assumed to play a paramount role. These datasets comprise meticulously curated collections of network traffic data, thoughtfully annotated to demarcate instances of normal network behavior from those tainted by nefarious activities. The objective behind the creation of IDS datasets is to facilitate the rigorous evaluation of IDS algorithms, thereby gauging their efficacy in detecting an array of malicious activities, encompassing but not limited to DDoS attacks, port scanning, and insidious malware infiltrations. The development of IDS datasets represents a multifaceted and labor-intensive endeavor, involving the intricate processes of data collection, judicious traffic filtration, meticulous labeling of distinct attack types, and the stringent preservation of data privacy and security considerations. Prominent organizations, such as the venerable National Institute of Standards and Technology (NIST), have undertaken the arduous task of formulating IDS datasets. These datasets stand as invaluable resources, widely embraced by the research community for their pivotal role in advancing the field of cybersecurity.

Access to high-quality IDS datasets stands as a linchpin in the development and evaluation of IDS algorithms. As underscored by Hnamte and Hussain (2021), these datasets serve as indispensable tools, empowering both researchers and practitioners to rigorously assess the performance of their algorithms across an expansive spectrum of attack scenarios. Beyond mere assessment, these datasets also function as treasure troves of information, enabling the discernment of patterns and trends within network attacks. Such insights are instrumental in the perpetual quest to fortify network security defenses. However, it's noteworthy that until the introduction of InSDN dataset (Elsayed et al., 2020), there has been a noticeable dearth of datasets meticulously tailored for the SDN environment. The emergence of InSDN represents a pivotal milestone, addressing this critical gap in the research landscape

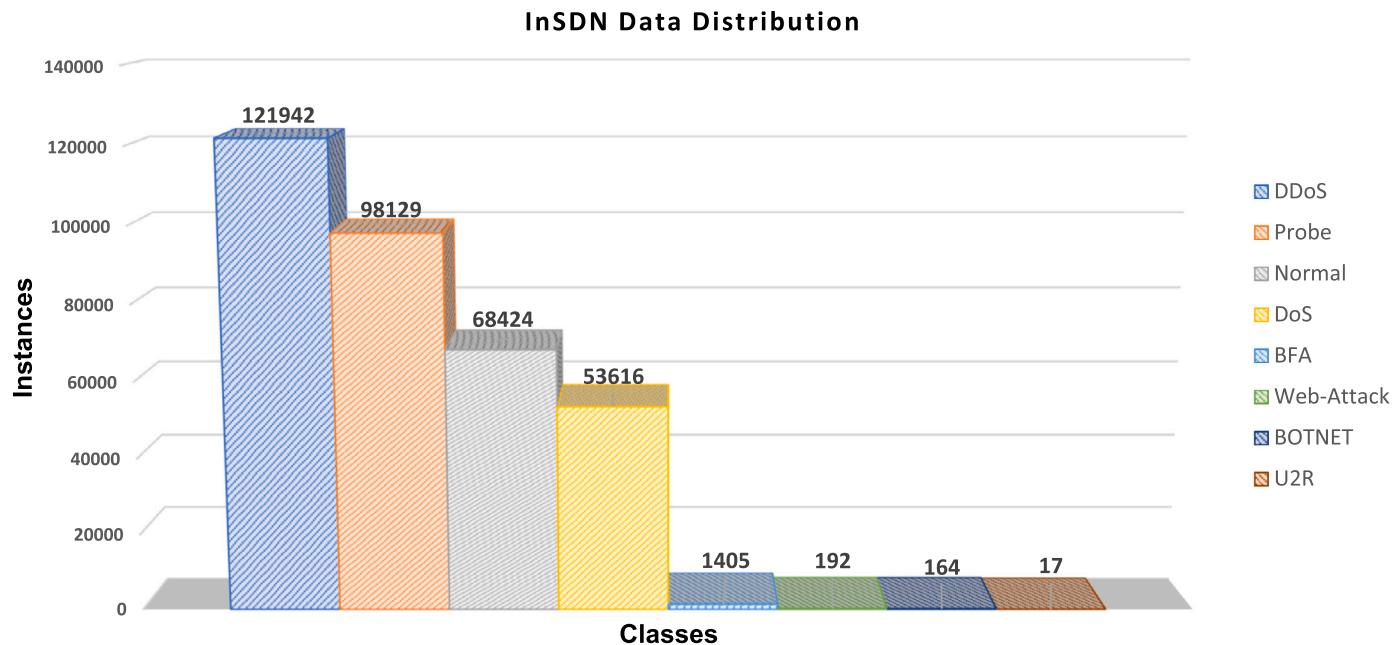


Fig. 3. InSDN data distribution. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

and heralding a new era of exploration and innovation in SDN-based network security.

4.1.1. InSDN

The InSDN² dataset was recently developed to address the limitations of existing datasets for DDoS attack detection in SDN environments. The InSDN dataset was created by collecting real-world network traffic from an SDN environment and labeling it according to the presence of DDoS attacks. Several virtual machines with an SDN network architecture were constructed to generate the dataset. The standard Ubuntu system represents regular users, whereas the Kali system represents attackers doing various forms of attacks on the SDN network. The dataset comprises a total of 343889 instances of data, with 84 characteristics per instance. The dataset contains eight separate traffic classes. There are 68424 instances of regular traffic and 275465 instances of attack traffic. Fig. 3 illustrates the distribution of data sets. The dataset includes a variety of attack scenarios, including TCP, UDP, and ICMP floods, as well as SYN floods and Slowloris attacks. For this reason, we have used the InSDN dataset to train the proposed DNN model.

Several modern IDSs for SDN use datasets such KDD99,³ NSL-KDD (Tavallaei et al., 2009), etc. These are good datasets; however, the protocol and network topology used by conventional networks are significantly different from those of SDN networks.

4.1.2. CICIDS2018

The CICIDS2018 (Sharafaldin et al., 2018) (Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset 2018) is a comprehensive and widely used benchmark dataset designed for the evaluation of IDS. Developed by the Canadian Institute for Cybersecurity, this dataset is instrumental in advancing research and development in the domain of network security. CICIDS2018 encompasses a diverse range of network traffic scenarios, including normal activities and various types of cyber attacks. The dataset is designed to simulate real-world network environments, capturing both benign and malicious network behaviors. The dataset covers a variety of network traffic types, such as normal traffic, DDoS, brute-force attacks, and more. Different attack

scenarios are represented, providing a holistic view of potential security threats.

Within the expansive CICIDS2018 dataset, the subset labeled “DDoS-LOIC-UDP-HOIC-21-02-2018” holds particular significance in the context of this research. Fig. 4 illustrates the distribution of data sets. Focused on instances of DDoS attacks orchestrated through LOIC, and HOIC, this subset provides a concentrated collection of network traffic data specifically related to DDoS scenarios. CICIDS2018 contains a substantial volume of network traffic data, enabling robust analysis and evaluation of intrusion detection algorithms. The dataset is sufficiently large to support the training and validation of DL models. The dataset is designed to mirror real-world network conditions, facilitating the creation and assessment of intrusion detection solutions that can be deployed in practical settings. Due to its scope, realism, and comprehensive labeling, the CICIDS2018 dataset has become a valuable resource in the cybersecurity research community. Researchers leverage this dataset to benchmark and compare the performance of various intrusion detection algorithms, ultimately contributing to the advancement of cybersecurity measures and practices.

4.1.3. Kaggle DDoS

In the realm of DDoS detection research, the unavailability of a perfectly tailored public dataset exclusively dedicated to DDoS scenarios has prompted researchers to employ innovative methodologies. Addressing this challenge, Prasad et al. (2019) undertook a meticulous process to curate a Kaggle DDoS dataset, amalgamating data from various public IDS datasets, namely CSE-CIIDS2018, CICIDS2017, and CICDoS 2016 datasets. Each of these IDS datasets provides a distinct perspective on DDoS activities, incorporating diverse attack scenarios and traffic patterns.

The extracted DDoS flows are not limited to a single temporal snapshot. Instead, data is drawn from IDS datasets produced in different years, offering a longitudinal perspective on the evolution of DDoS attacks. Additionally, the inclusion of DDoS flows generated using different experimental traffic generation tools enhances the dataset's comprehensiveness.

The inclusion of “Benign” flows is pivotal as it provides a baseline for comparison, enabling a more nuanced understanding of DDoS-induced anomalies. This comparative analysis is essential for the development of effective intrusion detection models capable of discerning malicious

² <https://aseados.ucd.ie/datasets/SDN/>.

³ <http://kdd.ics.uci.edu/>.

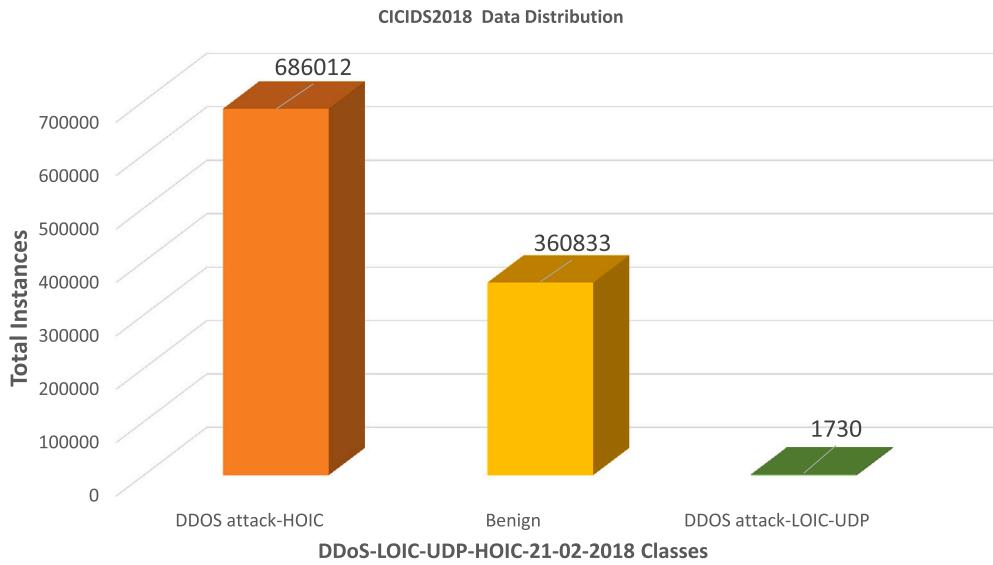


Fig. 4. CICIDS2018 data distribution.

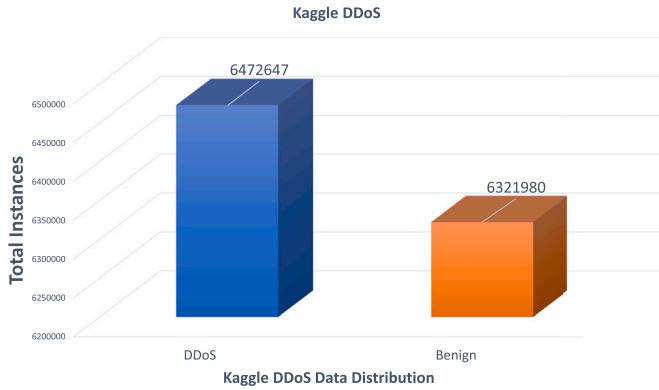


Fig. 5. Kaggle DDoS data distribution.

activities from routine network behavior. Fig. 5 illustrate the data distribution of Kaggle DDoS. The Kaggle_DDoS⁴ dataset emerges as a strategic response to the limitations in the availability of dedicated DDoS datasets.

4.2. Evaluation metrics

The efficacy of our proposed model is meticulously assessed through a comprehensive set of performance metrics, each offering distinct insights into the model's capabilities. The chosen metrics, namely precision, recall, F1-Score, and accuracy, are derived from a fundamental set of measures: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). This evaluation framework is widely acknowledged in the academic domain and has been extensively utilized to gauge the performance of various models in diverse contexts (Powers, 2011). The essential performance metrics are computed as follows:

$$TPR = \frac{TP}{TP + FN} \quad (1)$$

$$FPR = \frac{FP}{FP + TN} \quad (2)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{F1-Score} = 2 \times \frac{(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (5)$$

$$\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP} \quad (6)$$

Here, TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively. These metrics collectively form a robust framework for evaluating the model's performance across different facets.

In binary classification, the True Positive Rate (TPR), also known as sensitivity or recall, and the False Positive Rate (FPR) are pivotal metrics for assessing the effectiveness of a classification model. TPR is a measure of the model's capability to correctly identify instances belonging to the positive class. Whereas, FPR gauges the ratio of instances incorrectly classified as positive to the total number of actual negatives.

The equations encapsulate the intricate relationships among these metrics, providing a quantitative basis for assessing the model's precision, recall, overall predictive power (F1-Score), and classification accuracy. This evaluation methodology, based on foundational metrics, not only facilitates a nuanced understanding of the model's strengths and limitations but also aligns with established practices in the field of model evaluation (Powers, 2011).

The experimental findings derived from the proposed DNN model across various datasets, namely CICIDS2018, Kaggle DDoS, and InSDN, underscore a robust and formidable performance in the realm of DDoS attack detection. The model, trained over a course of 30 epochs, manifests commendable proficiency in discerning intricate patterns within each dataset, attaining a noteworthy level of accuracy and minimizing the associated loss.

4.2.1. CICIDS2018 dataset

- Training Dynamics:** The model exhibits an exemplary capacity to assimilate the intricacies of the CICIDS2018 dataset, achieving a flawless accuracy of 100% on the training set. The concomitant diminution of the loss function over the epochs underscores the model's adeptness in learning the underlying representations embedded in the training data.

- Validation Proficiency:** Generalization to the validation set is discerned through an impeccable accuracy of 100%, reinforcing the model's ability to transcend the confines of the training set.

⁴ <https://www.kaggle.com/datasets/devendra416/ddos-datasets/data>.

Table 3
Performance comparison across datasets.

Dataset	Evaluation Metrics					Time (in seconds)	
	Accuracy	Loss	Recall	Precision	F1 Score	Training	Inference
InSDN	99.98%	0.0052	99.98%	99.97%	99.97%	4.00 s	3.37 s
CICIDS2018	100%	0.0000	100%	100%	100%	15.00 s	9.75 s
Kaggle DDoS	99.99%	0.0000	99.99%	99.99%	99.99%	171.00 s	117.09 s

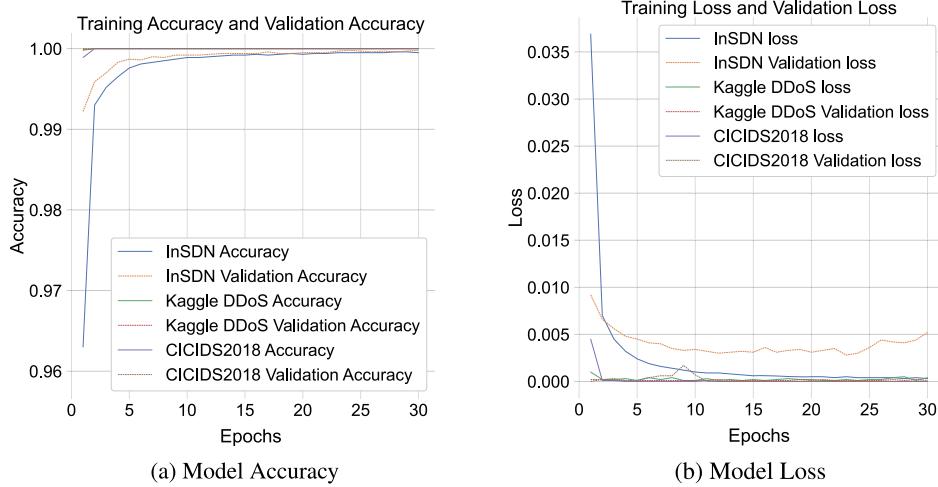


Fig. 6. Model accuracy and loss.

The nominal validation loss further corroborates the model's proficiency in extending its predictive capabilities to unseen instances.

- **Evaluation Metrics:** The suite of evaluation metrics, comprising recall, precision, F1 score, training time and inference time, uniformly attains a perfect score of 100%, reflecting a model that excels in both sensitivity and specificity, substantiating its efficacy in DDoS detection within the CICIDS2018 context.

4.2.2. Kaggle DDoS dataset

- **Training Prowess:** Parallel to the performance on the CICIDS2018 dataset, the DNN model demonstrates remarkable training accuracy of 100%, coupled with a rapid attenuation of the loss function, indicative of a model that rapidly adapts to the nuanced characteristics of the Kaggle DDoS dataset.
- **Validation Robustness:** The model sustains an elevated validation accuracy of 99.99%, accompanied by a notably low validation loss. This attests to the model's ability to generalize effectively to instances beyond the training data, thereby fortifying its reliability in practical deployment scenarios.
- **Evaluation Metrics:** The evaluation metrics, while not reaching a perfect score, maintain a marginal differential, substantiating the model's exceptional performance in precision, recall, F1 score, training time and inference time, consolidating its position as an adept classifier in the Kaggle DDoS domain.

4.2.3. InSDN dataset

- **Training Competence:** The DNN model imparts its discriminative capacity to the InSDN dataset, culminating in a commendable training accuracy of 99.98%, accompanied by a gradual attenuation of the loss function. This underscores the model's acumen in assimilating the distinctive features characterizing the InSDN data distribution.
- **Validation Excellence:** Validation accuracy of 99.98%, coupled with a judiciously low validation loss, attests to the model's adeptness in extrapolating learned patterns to previously unseen instances within the InSDN dataset.

- **Evaluation Metrics:** While maintaining superlative performance across evaluation metrics, it is noteworthy that the training time and inference time, while marginally reduced compared to the other datasets, is indicative of the model's capability to delineate between benign and malicious activities within the InSDN paradigm.

The DNN model evinces a consistent and formidable performance across diverse datasets, substantiating its mettle as a robust classifier in the domain of DDoS attack detection. The convergence of high accuracy, diminished loss, and superlative evaluation metrics collectively validate the efficacy of the model in real-world scenarios, portraying it as a potent tool for bolstering network security through the identification of DDoS threats. The detailed results of the model performance, including accuracy, loss, evaluation metrics (recall, precision, F1-Score), inference time, and training time, across different datasets are presented in Table 3.

Fig. 6 presents a comprehensive depiction of the training and validation performance metrics across multiple datasets, namely InSDN, Kaggle DDoS, and CICIDS2018. The figure provides a detailed insight into the model's learning dynamics and generalization capabilities.

For each dataset, the training accuracy, validation accuracy, training loss, and validation loss are illustrated over the course of training epochs in Fig. 6a and 6b. These metrics serve as crucial indicators of the model's ability to learn from the training data and its performance on previously unseen validation data.

The training accuracy curve showcases the model's ability to correctly classify instances within the training dataset, while the training loss curve reflects the convergence of the model during training. Simultaneously, the validation accuracy and validation loss curves offer insights into the model's performance on data not used during training, thereby indicating its generalization capabilities.

The distinct trajectories of these curves for each dataset highlight the dataset-specific intricacies in model learning and performance. Analyzing these curves aids in assessing how well the model adapts to the characteristics of each dataset and whether it is prone to overfitting or underfitting.

Table 4
Comparison of models performance.

Reference	Model	Dataset	Evaluation Metrics		Time (in seconds)	
			Accuracy	Loss	Training	Inference
AL-Hawawreh et al. (2018)	ADS	NSL-KDD	92.4%	8.2	194.67	5.55 s
		UNSW-NB15	98.6%	1.8	119.03	2.25 s
Elmasry et al. (2020)	PSO + DNN	NSL-KDD	96.38%	0.51	*	*
	PSO + LSTM-RNN		98.18%	0.39	*	*
	PSO + DBN		99.81%	0.23	*	*
	PSO + DNN	CICIDS2017	97.58%	0.28	*	*
	PSO + LSTM-RNN		98.68%	0.16	*	*
	PSO + DBN		99.92%	0.1	*	*
Marvi et al. (2021)	LGBM/PM-Model	CICDDoS2019	100%	0.0038	<3 mins	*
	LGBM/LDAP-Model		99.98%	0.0056	<5 mins	*
	LGBM/SYN Model		99.98%	0.0023	<10 mins	*
Cil et al. (2021)	DNN	CICDDoS2019_1	99.97%	*	*	*
		CICDDoS2019_2	94.57%	*	*	*
Hnamte and Hussain (2023c)	DCNN	ISCX2012	99.79%	0.0058	26 s	19.50 s
		Kaggle DDoS - 1	99.99%	0.0002	10 s	7.11 s
		Kaggle DDoS - 2	100%	0.0000	17 s	11.79 s
		CICIDS2017	99.96%	0.0015	40 s	29.36 s
		CICIDS2018	100%	0.0000	15 s	9.91 s
Hnamte and Hussain (2023b)	DCNNBiLSTM	CICIDS2018	100%	0.0000	202 s	95.04 s
		EDGE_IoT	99.64%	0.0080	500 s	219.29 s
<i>This study</i>	DNN	InSDN	99.98%	0.0052	4.00 s	3.37 s
		CICIDS2018	100%	0.0000	15.00 s	9.75 s
		Kaggle DDoS	99.99%	0.0000	171.00 s	117.09 s

* Not available.

Fig. 6 serves as a valuable reference for researchers and practitioners seeking a nuanced understanding of the model's behavior across diverse datasets, contributing to the interpretability and generalization analysis of the trained model.

Table 3 provides a comprehensive overview of the quantitative performance metrics for the employed model across distinct datasets, including InSDN, Kaggle DDoS, and CICIDS2018. This table encapsulates key evaluation metrics such as recall, precision, F1 score, accuracy, loss, inference time, and training time, facilitating a detailed comparison of the model's performance on each dataset.

In conjunction with the detailed numerical results presented in Table 3, Fig. 6 serves as a visual counterpart, offering a dynamic portrayal of the model's learning and generalization patterns during training epochs. This figure encompasses training accuracy, training loss, validation accuracy, and validation loss curves for each dataset, providing insights into the model's behavior over time. Researchers and practitioners can synergistically utilize both resources to draw nuanced conclusions about the model's strengths, weaknesses, and its adaptability to distinct data characteristics.

In the context of deploying DL models for intrusion detection within an SDN environment, each of the models, trained on distinct datasets comprising InSDN, Kaggle DDoS, and CICIDS2018, holds potential applicability. However, in pursuit of operational simplicity, a judicious decision has been made to select the model derived from the InSDN dataset, primarily owing to its marginally diminished accuracy performance relative to its counterparts. This deliberate choice is motivated by the overarching goal of optimizing the intricacies associated with model integration into the SDN infrastructure while still harnessing the discerning capabilities of the chosen model for robust detection of DDoS attacks within the SDN paradigm. This strategic alignment aims to balance model efficacy with pragmatic considerations, ensuring a streamlined and effective deployment process tailored to the unique demands of the SDN environment.

Table 4 provides a comparative analysis of various models employed for DDoS attack detection, including the current study. Each entry delineates the reference, model architecture, dataset used, evaluation metrics (accuracy and loss), and time-related metrics (training and inference durations). In comparison to the referenced studies, the proposed DNN model in this study demonstrates competitive or superior accuracy across all datasets. Moreover, the training and inference times are considerably lower, indicating the model's efficiency. It's noteworthy that the DNN architecture employed in this study achieves comparable or better results with reduced computational overhead, making it a promising approach for real-time DDoS attack detection in SDN environments.

The efficacy of the DNN model in detecting DDoS attacks has been empirically validated, establishing its practical applicability. With this demonstrated capability, the next imperative step involves the implementation of DDoS attack detection within the SDN paradigm. In pursuit of this objective, our chosen approach involves the utilization of a pre-trained DNN model. This model has been previously trained using the InSDN dataset, a comprehensive and representative dataset for SDN environments.

The integration of the DNN model into the SDN framework is orchestrated through the Ryu controller. The Ryu controller serves as a pivotal component, facilitating seamless communication and coordination between the DNN model and the SDN infrastructure. This strategic amalgamation aims to harness the discriminative capabilities of the DNN model to discern normal network behavior from anomalous patterns indicative of potential DDoS attacks. The utilization of a pre-trained model enhances the efficiency and accuracy of DDoS detection within the SDN ecosystem, offering a sophisticated layer of defense against evolving cyber threats.

4.3. System setup

The system configuration, delineated in the following, establishes the foundational framework for conducting experimental simulations in the realm of SDN. Each parameter in the table contributes to the nuanced configuration and distinctive characteristics of the simulated network. The elucidation of the system setup is as follows:

- **Host:** The chosen operating system is Ubuntu 22.04.
- **Python:** Version 3.9.5 of the Python programming language is employed for scripting and executing diverse functionalities within the system.
- **CPU/RAM:** The hardware infrastructure comprises an Intel i9 10900K processor and 64 GB of DDR4 RAM.
- **Emulator:** Mininet (<https://mininet.org/>), a network emulator, is harnessed to instantiate a virtual SDN environment, providing a controlled space for testing and developmental endeavors.
- **Controller:** Ryu is designated as the SDN controller, assuming responsibility for the orchestration and regulation of network behaviors.
- **Number of Controllers:** The system integrates two Ryu controllers to augment the managerial and control capabilities of the SDN.
- **Number of Switches:** Six switches are incorporated into the network configuration to facilitate the routing and forwarding of network traffic.
- **Number of Hosts:** Each switch accommodates three hosts, aggregating to a total of three hosts in each of the six switches.
- **Protocol:** OpenFlow serves as the communication protocol between the SDN controller and the switches.
- **Visualization:** MiniEdit is employed as a visualization tool, offering graphical representation of the network topologies and configurations.
- **Traffic Generation:** The mgen (<https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/MGEN/>) tool is employed for the purpose of generating controlled network traffic within the SDN.
- **Port for Controller:** The controllers utilize ports 6633 and 6634 for communication with the switches.
- **Simulation Duration:** The simulation is conducted over a duration of 5 hours.
- **Statistics Collection Interval:** Network statistics are systematically collected at regular intervals of 40 seconds throughout the simulation.
- **Bandwidth Plot Interval:** Bandwidth plots are generated at intervals of 30 seconds, contributing to the comprehensive assessment of network performance.

- **Number of Topologies:** The experimentation encompasses the consideration of six distinct network topologies, each contributing to the diverse scenarios evaluated within the study.

4.4. Network topologies, attacks detection and mitigation

In the context of SDN, network topologies serve as the architectural foundation, defining the interconnection of switches, controllers, and hosts within a given network fabric. This infrastructure is pivotal for orchestrating the communication and behavior of network components. The typical constituents include switches forming the network fabric, controllers managing switch behavior, and hosts facilitating communication through the network. The SDN topology can manifest diverse configurations, ranging from simple structures to intricate arrangements, such as trees, meshes, or hybrids, depending on the specific demands of the network.

In our experimental setup, a network controller instantiated through the Ryu framework assumes the crucial role of orchestrating an SDN network topology. This topology encompasses six switches (switch_1 to switch_6) and hosts (host_1 to host_18). The controller establishes communication with switches, exerting control over packet flow in the SDN infrastructure.

The network controller, implemented as a Ryu application, adeptly handles events and messages from network switches. By defining specific flow rules, the controller governs switch routing, ensuring the directed flow of packets throughout the network in adherence to predefined specifications. The ensuing code snippet illustrates the instantiation and configuration of this network.

The pivotal function, `get_topology_data`, assumes the responsibility of simulating DDoS attacks by generating a substantial volume of packet-in events. Each packet-in event is emblematic of an attack packet, complete with a randomly assigned source IP address.

In addition, the `packet_in_handler` function has been meticulously adapted to address these attack packets. Researchers and practitioners can, at their discretion, devise bespoke logic within this function to process the attack packets and execute appropriate actions. These actions may encompass traffic blockade or the initiation of alert mechanisms to signal potential security breaches.

As shown in the following code snippet, the app will generate a large number of packet-in events, running in endless loop, simulating a DDoS attack. The packet-in events will trigger the `packet_in_handler` function, allowing you to implement the desired DDoS detection and mitigation mechanisms.

```
# Define the DNN model path
MODEL_PATH = 'path_to_saved_model.h5'

class DDoSDetectionController(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(DDoSDetectionController, self).__init__(*args, **kwargs)

        # Load the saved DNN model
        self.dnn_model = tf.keras.models.load_model(MODEL_PATH)

    @set_ev_cls(event.EventSwitchEnter)
    def get_topology_data(self, ev):
        # Simulate DDoS attack by generating a large number of packet-in events
        num_attacks = 1000 # Number of packet-in events to generate

        for _ in range(num_attacks):
```

```

# Generate random source IP address for the attack packet
src_ip = f"{{random.randint(1, 255)}.{random.randint(1, 255)}.
{random.randint(1, 255)}.{random.randint(1, 255)}"

# Create the attack packet
pkt = packet.Packet()
pkt.add_protocol(ethernet.ethernet(ether_type=ether_types.ETH_TYPE_IP))
pkt.add_protocol(ipv4.ipv4(src=src_ip, dst='1.0.0.4'))

# Create a packet-in event
msg = ev.msg
msg.data = pkt.serialize()
msg.match = self.create_match()
msg.buffer_id = 0xffffffff

# Call the packet_in_handler to process the attack packet
self.packet_in_handler(msg)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    msg = ev.msg
    datapath = msg.datapath
    parser = datapath.ofproto_parser

    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        return

    # Extract relevant features from the packet
    src_ip = pkt.get_protocol(ipv4.ipv4).src
    dst_ip = pkt.get_protocol(ipv4.ipv4).dst
    src_port = pkt.get_protocol(tcp.tcp).src_port
    dst_port = pkt.get_protocol(tcp.tcp).dst_port

    # Preprocess the features
    features = np.array([src_ip, dst_ip, src_port, dst_port])
    preprocessed_features = preprocess_features(features)

    # Pass the preprocessed features to the DNN model for prediction
    prediction = self.dnn_model.predict(preprocessed_features)

    # Check if the prediction indicates a DDoS attack
    if prediction > 0.5:
        # Block the incoming packet from the source
        self.logger.info("DDoS attack detected: Packet from %s to %s", src_ip, dst_ip)

        # Block the incoming packet from the source
        self.block_source_packet(datapath, msg, src_ip)

        # Add the source IP address to the spoof IP table
        self.add_to_spoof_ip_table(src_ip)

def preprocess_features(self, features):
    # Extract features from the raw feature vector
    src_ip, dst_ip, src_port, dst_port = features

    # Normalize numerical features
    src_port_normalized = self.normalize_port(src_port)
    dst_port_normalized = self.normalize_port(dst_port)

    # Convert IP addresses to numerical values
    src_ip_numeric = self.ip_to_numeric(src_ip)
    dst_ip_numeric = self.ip_to_numeric(dst_ip)

```

```

# Combine all preprocessed features into a single array
preprocessed_features = np.array([src_ip_numeric, dst_ip_numeric,
src_port_normalized, dst_port_normalized])

return preprocessed_features

def normalize_port(self, port):
    # Normalize port to a range between 0 and 1
    return port / 65535.0 # Assuming port ranges from 0 to 65535

def ip_to_numeric(self, ip_address):
    # Convert IP address to a numerical value
    ip_parts = [int(part) for part in ip_address.split('.')]
    return sum(ip_parts[i] << (24 - 8 * i) for i in range(4))

def block_source_packet(self, datapath, msg, src_ip):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    # Blocking incoming packets from the source IP
    match = parser.OFPMatch(eth_type=0x0800, ipv4_src=src_ip)
    actions = []

    # Drop the packet
    inst = [parser.OFPIInstructionActions(ofproto.OFPIT_CLEAR_ACTIONS, actions)]
    mod = parser.OFPFlowMod(datapath=datapath, table_id=0, priority=10, match=match,
    instructions=inst)
    datapath.send_msg(mod)

    # Log the information
    self.logger.info("Blocked incoming packets from source IP: %s", src_ip)

def add_to_spoof_ip_table(self, src_ip):
    # Add the source IP to your spoof IP table
    # You need to implement this method based on your application's data structure
    # It might involve updating a list, dictionary, or database
    pass

```

The provided code represents a Ryu application designed for the detection of DDoS attacks in a SDN environment. The underlying mechanism relies on a pre-trained DNN model for predicting potential DDoS attacks based on extracted features from incoming network packets.

The necessary libraries are also imported, including numpy, tensorflow, and the required Ryu modules. The path to the saved DNN model is defined in the MODEL_PATH variable. The DDoSDetectionController class is defined, which extends the RyuApp class provided by Ryu. The `_init_` method initializes the controller and loads the saved DNN model using `tf.keras.models.load_model()`.

The process begins with the instantiation of the SDN topology, where a simulated DDoS attack is emulated by generating a substantial number of packet-in events. Each event encapsulates an attack packet with randomly assigned source IP addresses. The `packet_in_handler` method processes these events, extracting relevant features from the incoming packets, preprocesses these features, and passes them through the pre-trained DNN model for prediction.

Upon prediction, if the model determines a likelihood of a DDoS attack ($\text{prediction} > 0.5$), the system logs the pertinent information, signaling the detection of a potential threat. At this juncture, the code provides the necessary actions for DDoS attack mitigation.

In the context of detection accuracy, the model's proficiency in identifying DDoS attacks can be ascertained through the evaluation of the prediction accuracy metric, which is not explicitly presented in the provided code snippet. However, assuming a well-trained DNN model which was shown earlier in the previous subsection, the detection accu-

racy would be contingent upon the model's ability to generalize effectively to unseen data.

From a comparative standpoint, the code alludes to the ease of implementing mitigation measures in response to identified DDoS attacks within an SDN framework. The inherent programmability and dynamic control afforded by SDN, coupled with the capabilities of the Ryu controller, facilitate swift and adaptive responses to security incidents. This stands in contrast to traditional networking paradigms where manual reconfiguration and static rule-based approaches may impede the agility required for timely threat mitigation.

4.5. Performance scenario

The attacker gains control over one of the hosts and uses it to launch a DoS attack against another host (the victim). The legitimate traffic comes from a different host, which is neither the victim nor the attacker, and this legal host tries to communicate with the controller in a harmless manner. Despite the fact that the topology created is not a large one, the attacker can generate malicious packet streams that appear to originate from various IP addresses. While the topology is not extensive, the information collected will be helpful in comprehending how DoS attacks function on small to medium-sized networks and will serve as the foundation for future research in this area.

```

# Using scapy tools to send normal traffic
>>> p = IP(dst="1.0.0.9")/ICMP()
>>> r = sr1(p)

```

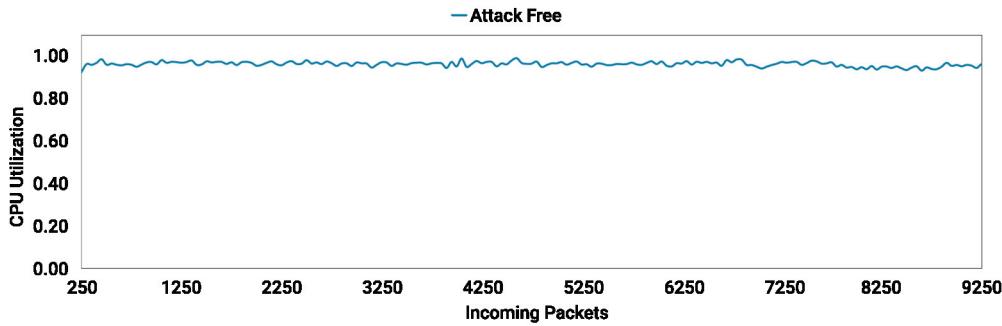


Fig. 7. CPU utilization: packets in an attack-free scenario.

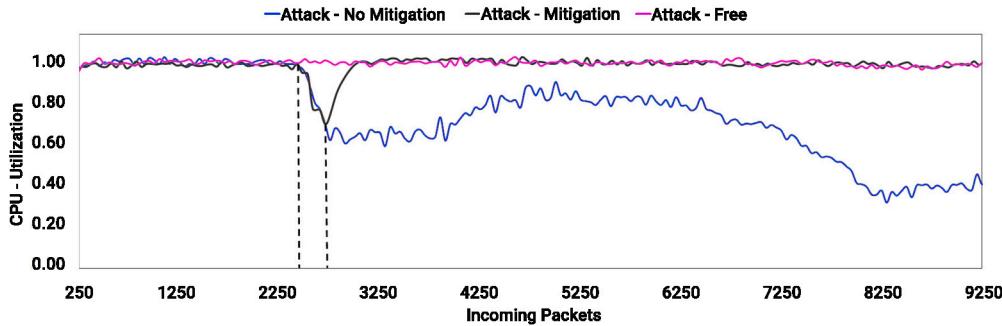


Fig. 8. CPU utilization: packets in an under-attack scenario.

```

Begin emission:
Finished to send 1 packets.
.
.
.
>>> r[IP].src
'10.0.0.4'

```

At first, the performance of the CPU is evaluated when the network is not under attack and generates normal traffic using the Scapy tool.⁵ In this scenario, the hosts transmit packets with equal likelihood, which indicates that there is no clustering of traffic around a specific target. Fig. 7 illustrates the connection between the number of incoming packets received by the controller when the time interval for generating traffic is set to 0.02 seconds. This means that within 0.02 seconds, a batch of packets is generated and sent to the controller from the authentic host.

The utilization of the CPU in relation to the number of incoming "packet_in" requests is depicted in Fig. 8, both with and without the proposed method under the attack scenario. The measurements shown in the figure were taken after the system was established, following the completion of the exchange of ARP requests and replies between the hosts during the initial network setup. During this experiment, the interval for generating regular traffic was set at 0.02 seconds, while the interval for generating attacking traffic was set at 0.005 seconds, which translates to 25% of the traffic being malicious. The Scapy tool was used to generate attacking traffic, with the source IP being spoofed and directed towards the victim host.

When a host receives a large volume of packets, the system's randomness decreases quickly, causing the CPU utilization to drop below the threshold and allowing for the maximum available CPU resource. This is because all the packets with a spoofed source IP have the victim's IP address in their destination IP, resulting in less variability in destination addresses. This inverse relationship between IP address variety

and CPU utilization is significant. It's worth noting that if the proposed method was not used, the CPU resources would have initially dropped, but then there would have been some slight enhancements. This improvement would have been possible because of the partial failure of the controller due to high traffic volume, leading to dropped malicious requests or disconnecting the OpenFlow switch to which the attacker is connected. Other reasons could include the overload of the OpenFlow switch tables and the flow rules installed by the controller on the OpenFlow switches, reducing the number of incoming "packet_in" requests from the attacker to the controller. As the attacking traffic continues to be directed towards the controller, the entropy value gradually drops until the end of the attack period. Finally, as the system receives genuine packets after the attack is over, it improves.

We conducted an experiment to assess the effectiveness of the proposed DNN in detecting attacks of varying intensities. The experiments involved generating attack traffic at different rates and measuring the time it took for the DNN to detect the attack. The regular traffic generation interval was kept constant at 0.02 seconds. Fig. 9 shows that higher attack rates led to shorter detection times, likely because the entropy value dropped rapidly in a short period of time. Increasing the attack intervals resulted in a greater number of attacking packets being generated. Fig. 9 shows that when the attack rate is higher, the detection time is shorter. This is because the entropy value drops quickly within a short period of time.

In a different series of experiments, the team evaluated the effectiveness of the proposed solution in maintaining reliable network connections during an attack. They used a packet analyzer tool, Wireshark, to analyze the network's behavior by examining the exchanged packets. In particular, they filtered out the TCP packets marked for re-transmission, which are packets that are resent due to network congestion or partial failure.

The results are visually presented in Fig. 10, illustrating the TCP packets that were exchanged and subsequently marked for re-transmission within distinct time frames, during both attack-free and under-attack conditions. Initially, the pattern of packet exchange exhibits similarities in both scenarios. However, a notable deviation becomes evident as the attacker instigates a substantial volume of

⁵ <https://github.com/scapy/scapy/releases>.

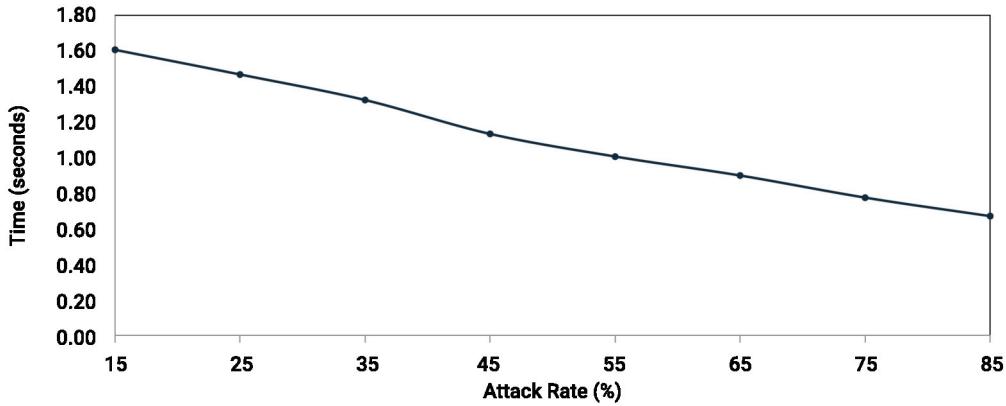


Fig. 9. Attack detection time versus attack rate.

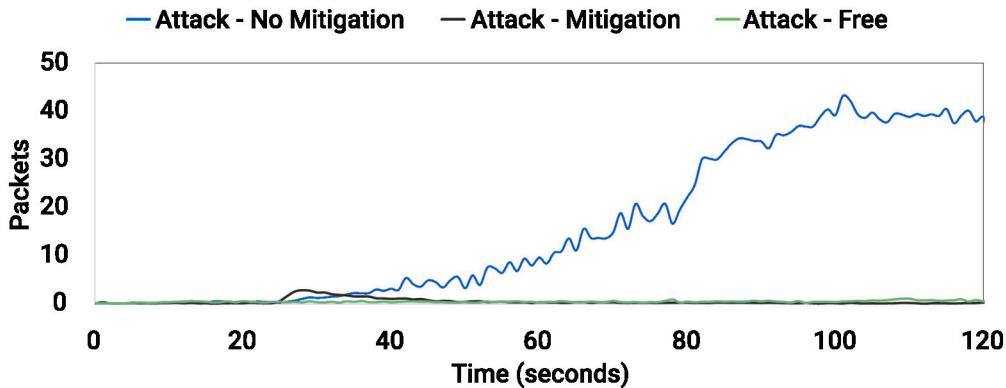


Fig. 10. Exchanged control packets over time.

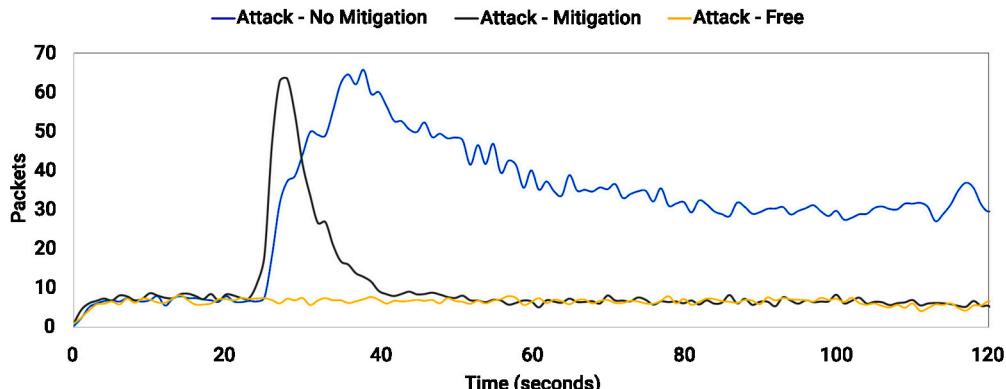


Fig. 11. Incoming packets to victim from all sources over time.

"packet_in" requests, culminating in a noteworthy surge in the count of packets marked for re-transmission approximately at $t = 25$ seconds. This conspicuous escalation signifies instances of packet loss or corruption within the network, thereby contributing to heightened latency and a reduction in request processing velocity.

Furthermore, we conducted an assessment of the effectiveness of the proposed method in shielding the victim host against incoming attack packets. To conduct this evaluation, we conducted an analysis of all packets that were successfully received by the victim host. Fig. 11 provides an inclusive representation of all packets directed towards the victim host, encompassing those originating from both legitimate sources and malicious attackers. In contrast, Fig. 12 selectively presents only those packets initiated by attackers and explicitly aimed at the victim host. This selection was facilitated by employing the Wireshark

analyzer, utilizing the destination IP address of the victim host as a filter criterion.

At the beginning of the experiment, the trend in received traffic on the victim host was the same for both the attack-free and attack scenarios, with traffic from both legitimate and illegitimate sources being reported in Fig. 11. However, when the attacker begins to send fake packets at around $t = 25$ seconds, the number of packets received by the victim host starts to increase.

5. Conclusion

The persistent and escalating threat of DDoS attacks necessitates continuous advancements in defensive strategies to safeguard online services and applications. This manuscript introduces a robust and effective approach to DDoS attack detection and mitigation within SDN

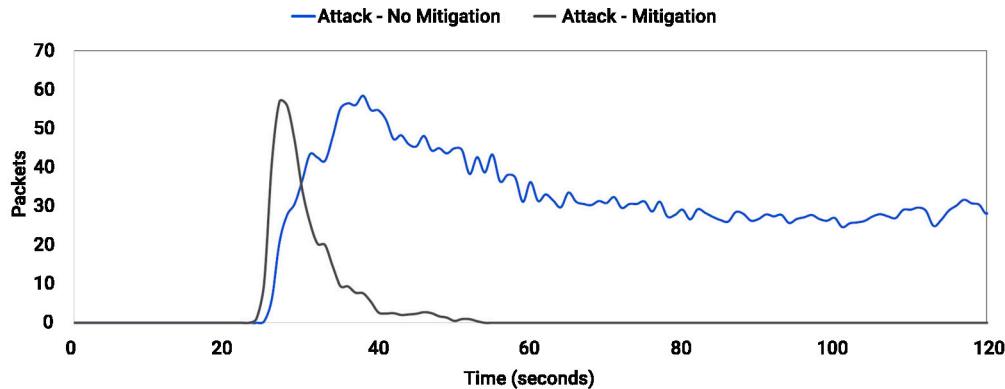


Fig. 12. Incoming packets to victim from attacker over time.

environments through the deployment of DNN-based detection systems. The proposed model has undergone meticulous validation using three diverse datasets — CICIDS2018, InSDN, and Kaggle DDoS datasets — affirming its efficacy in detecting and countering DDoS attacks, as evidenced in the results section.

The application of DNNs in identifying DDoS attacks within SDN environments has demonstrated remarkable potential. DNNs, with their intrinsic capability to meticulously scrutinize network traffic data, unveil intricate patterns indicative of DDoS attacks. This analytical prowess empowers organizations to respond swiftly and precisely to potential threats. The availability of high-caliber datasets, exemplified by the InSDN dataset, emerges as a cornerstone for the development and comprehensive evaluation of robust DNN-based DDoS attack detection systems.

Crucially, our results indicate that CPU availability during Packet-in events under an attack scenario remains almost the same as in free-attack scenarios. This underscores the efficiency and feasibility of the proposed DNN-based model, suggesting that it can handle DDoS attacks without compromising CPU resources, a vital aspect for maintaining network performance.

As DDoS attacks evolve in sophistication, future research should prioritize the refinement of DNN-based detection systems, enhancing their robustness and efficiency to cope with the escalating intricacy of contemporary DDoS attacks. The exploration of hybrid models, seamlessly integrating DNNs with complementary techniques such as traffic profiling or feature selection, holds significant promise for augmenting the precision and efficiency of DDoS attack detection.

Additionally, future research directions should focus on enhancing the scalability and adaptability of DNN-based detection systems to operate effectively within large-scale SDN networks. The development of real-time DNN-based DDoS attack detection systems is also imperative, requiring the innovation of avant-garde algorithms and architectural constructs to operate seamlessly in real-time environments with stringent latency constraints.

Lastly, prospective research endeavors should meticulously evaluate DNN-based detection systems across diverse SDN environments and network topologies, including edge computing, Quality of Service (QoS) paradigms, and IoT networks. Holistic assessments in these domains promise a deeper comprehension of the challenges and attributes of SDN-based network security, facilitating the development of bespoke and laser-focused defense mechanisms against the ever-evolving landscape of DDoS attacks. The marriage of advanced technologies and insightful research endeavors holds the promise of fortifying defenses against the relentless onslaught of DDoS threats in the future.

CRediT authorship contribution statement

Vanlalruata Hnamte: Data curation, Writing – original draft, Conceptualization, Methodology, Formal analysis. **Ashfaq Ahmad Najar:**

Visualization, Literature review, Conceptualization, Validation, Formal analysis. **Hong Nhung-Nguyen:** Writing – review and editing. **Jamal Hussain:** Supervision, resources. **S. Manohar Naik:** Supervision, resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

- Agarwal, A., Khari, M., Singh, R., 2021. Detection of DDoS attack using deep learning model in cloud storage application. *Wirel. Pers. Commun.*, 1–21. <https://doi.org/10.1007/s11277-021-08271-z>.
- AL-Hawawreh, M., Moustafa, N., Sitnikova, E., 2018. Identification of malicious activities in industrial Internet of things based on deep learning models. *J. Inf. Secur. Appl.* 41, 1–11. <https://doi.org/10.1016/j.jisa.2018.05.002>.
- Ali, J., Roh, B.-h., Lee, B., Oh, J., Adil, M., 2020. A machine learning framework for prevention of software-defined networking controller from DDoS attacks and dimensional reduction of big data. In: 2020 International Conference on Information and Communication Technology Convergence (ICTC), pp. 515–519.
- Aljabri, M., Aljameel, S.S., Mohammad, R.M.A., Almotiri, S.H., Mirza, S., Anis, F.M., Aboulnour, M., Alomari, D.M., Alhamed, D.H., Altamimi, H.S., 2021. Intelligent techniques for detecting network attacks: review and research directions. *Sensors* 21 (21), 7070. <https://doi.org/10.3390/s21217070>.
- Amaizu, G., Nwakanma, C., Bhardwaj, S., Lee, J., Kim, D., 2021. Composite and efficient DDoS attack detection framework for B5G networks. *Comput. Netw.* 188, 107871. <https://doi.org/10.1016/j.comnet.2021.107871>.
- Banitalebi Dehkordi, A., Soltanaghaei, M., Boroujeni, F.Z., 2021. The DDoS attacks detection through machine learning and statistical methods in SDN. *J. Supercomput.* 77 (3), 2383–2415. <https://doi.org/10.1007/s11227-020-03323-w>.
- Berman, D.S., Buczak, A.L., Chavis, J.S., Corbett, C.L., 2019. A survey of deep learning methods for cyber security. *Information* 10 (4), 122. <https://doi.org/10.3390/info10040122>.
- Bhuyan, M., Kalwar, A., Goswami, A., Bhattacharyya, D., Kalita, J., 2015. Low-rate and high-rate distributed dos attack detection using partial rank correlation. In: 2015 Fifth International Conference on Communication Systems and Network Technologies, pp. 706–710.
- Chanu, U.S., Singh, K.J., Chanu, Y.J., 2023. A dynamic feature selection technique to detect DDoS attack. *J. Inf. Secur. Appl.* 74, 103445. <https://doi.org/10.1016/j.jisa.2023.103445>.
- Chouhan, R.K., Atulkar, M., Nagwani, N.K., 2023. A framework to detect DDoS attack in Ryu controller based software defined networks using feature extraction and classification. *Appl. Intell.* 53, 4268–4288. <https://doi.org/10.1007/s10489-022-03565-6>.
- Cil, A.E., Yildiz, K., Buldu, A., 2021. Detection of DDoS attacks with feed forward based deep neural network model. *Expert Syst. Appl.* 169, 114520. <https://doi.org/10.1016/j.eswa.2020.114520>.
- Elmasry, W., Akbulut, A., Zaim, A.H., 2020. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Comput. Netw.* 168, 107042. <https://doi.org/10.1016/j.comnet.2019.107042>.

- Elsayed, M.S., Le-Khac, N.-A., Jurcut, A.D., 2020. InSDN: a novel SDN intrusion dataset. *IEEE Access* 8, 165263–165284. <https://doi.org/10.1109/ACCESS.2020.3022633>.
- Fatani, A., Dahou, A., Al-Qaness, M.A., Lu, S., Abd Elaziz, M., 2022. Advanced feature extraction and selection approach using deep learning and Aquila optimizer for IoT intrusion detection system. *Sensors* 22 (1), 140. <https://doi.org/10.3390/s22010140>.
- Ferrag, M.A., Maglaras, L., Moschoyiannis, S., Janicke, H., 2020. Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. *J Inf. Secur. Appl.* 50, 102419. <https://doi.org/10.1016/j.jisa.2019.102419>.
- Fouladi, R.F., Ermis, O., Anarim, E., 2022. A DDoS attack detection and countermeasure scheme based on DWT and auto-encoder neural network for SDN. *Comput. Netw.* 214, 109140. <https://doi.org/10.1016/j.comnet.2022.109140>.
- Hnamte, V., Hussain, J., 2021. An extensive survey on intrusion detection systems: datasets and challenges for modern scenario. In: 2021 3rd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE), pp. 1–10.
- Hnamte, V., Hussain, J., 2023a. DDoS detection using hybrid deep neural network approaches. In: 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), pp. 1–8.
- Hnamte, V., Hussain, J., 2023b. DCNNBiLSTM: an efficient hybrid deep learning-based intrusion detection system. *Telemat. Inform. Rep.* 10, 100053. <https://doi.org/10.1016/j.teler.2023.100053>.
- Hnamte, V., Hussain, J., 2023c. Dependable intrusion detection system using deep convolutional neural network: a novel framework and performance evaluation approach. *Telemat. Inform. Rep.* 11, 100077. <https://doi.org/10.1016/j.teler.2023.100077>.
- Hnamte, V., Nhung-Nguyen, H., Hussain, J., Hwa-Kim, Y., 2023. A novel two-stage deep learning model for network intrusion detection: LSTM-AE. *IEEE Access* 11, 37131–37148. <https://doi.org/10.1109/ACCESS.2023.3266979>.
- Hussain, J., Hnamte, V., 2021a. Deep learning based intrusion detection system: modern approach. In: 2021 2nd Global Conference for Advancement in Technology (GCAT), pp. 1–6.
- Hussain, J., Hnamte, V., 2021b. Deep learning based intrusion detection system: software defined network. In: 2021 Asian Conference on Innovation in Technology (ASIAN-CON), pp. 1–6.
- Hussain, J., Hnamte, V., 2021c. A novel deep learning based intrusion detection system: software defined network. In: 2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), pp. 506–511.
- Isa, M.M., Mhamdi, L., 2020. Native SDN intrusion detection using machine learning. In: 2020 IEEE Eighth International Conference on Communications and Networking (ComNet). IEEE, pp. 1–7.
- Karan, B., Narayan, D., Hiremath, P., 2018. Detection of DDoS attacks in software defined networks. In: 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), pp. 265–270.
- Li, C., Wu, Y., Yuan, X., Sun, Z., Wang, W., Li, X., Gong, L., 2018. Detection and defense of DDoS attack-based on deep learning in openflow-based SDN. *Int. J. Commun. Syst.* 31 (5), e3497. <https://doi.org/10.1002/dac.3497>.
- Marvi, M., Arfeen, A., Uddin, R., 2021. A generalized machine learning-based model for the detection of DDoS attacks. *Int. J. Netw. Manag.* 31 (6), e2152. <https://doi.org/10.1002/nem.2152>.
- Mishra, A., Gupta, N., Gupta, B., 2023. Defensive mechanism against DDoS attack based on feature selection and multi-classifier algorithms. *Telecommun. Syst.* 82, 229–244. <https://doi.org/10.1007/s11235-022-00981-4>.
- Najar, A.A., Manohar Naik, S., 2022. DDoS attack detection using MLP and random forest algorithms. *Int. J. Inf. Technol.* 14 (5), 2317–2327. <https://doi.org/10.1007/s41870-022-01003-x>.
- Pérez-Díaz, J.A., Valdovinos, I.A., Choo, K.-K.R., Zhu, D., 2020. A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. *IEEE Access* 8, 155859–155872. <https://doi.org/10.1109/ACCESS.2020.3019330>.
- Powers, D., 2011. Evaluation: from precision, recall and f-measure to ROC, informedness, markedness & correlation. *J. Mach. Learn. Technol.* 2 (1), 37–63. <https://doi.org/10.48550/arXiv.2010.16061>.
- Prasad, M., Devendra, Babu V., Prasanta, Amarnath, C., 2019. Machine learning DDoS detection using stochastic gradient boosting. *Int. J. Comput. Sci. Eng.* 7, 157–166. <https://doi.org/10.26438/ijcse/v7i4.157166>.
- Said Elsayed, M., Le-Khac, N.-A., Dev, S., Jurcut, A.D., 2020. Network anomaly detection using LSTM based autoencoder. In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pp. 37–45.
- Santos-Neto, M.J., Bordim, J.L., Alchieri, E.A.P., Ishikawa, E., Dourado, L.S., 2022. Detecting DDoS attacks in SDN using a hybrid method with entropy and machine learning. In: 2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW), pp. 248–254.
- Sayed, M.S.E., Le-Khac, N.-A., Azer, M.A., Jurcut, A.D., 2022. A flow-based anomaly detection approach with feature selection method against DDoS attacks in SDNs. *IEEE Trans. Cogn. Commun. Netw.* 8 (4), 1862–1880. <https://doi.org/10.1109/TCCN.2022.3186331>.
- Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP. INSTICC, SciTePress, pp. 108–116.
- Swami, R., Dave, M., Ranga, V., 2023. Mitigation of DDoS attack using moving target defense in SDN. *Wirel. Pers. Commun.*, 1–15. <https://doi.org/10.1007/s11277-023-10544-8>.
- 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. In: 2016 International Conference on Wireless Networks and Mobile Communications (WIN-COM), pp. 258–263.
- Tavallaei, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6.
- Wei, Y., Jang-Jaccard, J., Sabrina, F., Singh, A., Xu, W., Camtepe, S., 2021. AE-MLP: a hybrid deep learning approach for DDoS detection and classification. *IEEE Access* 9, 146810–146821. <https://doi.org/10.1109/ACCESS.2021.3123791>.
- Yan, Q., Yu, F.R., Gong, Q., Li, J., 2016. Software-Defined Networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges. *IEEE Commun. Surv. Tutor.* 18 (1), 602–622. <https://doi.org/10.1109/COMST.2015.2487361>.
- Yuan, X., Li, C., Li, X., 2017. DeepDefense: identifying DDoS attack via deep learning. In: 2017 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, pp. 1–8.
- Zainudin, A., Ahakonye, L.A.C., Akter, R., Kim, D.-S., Lee, J.-M., 2022. An efficient hybrid-DNN for DDoS detection and classification in software-defined IIoT networks. *IEEE Int. Things J.*, 1. <https://doi.org/10.1109/JIOT.2022.3196942>.

Vanlaluarta Hnamte received a B.C.A in the year 2009 from Makhanlal Chaturvedi National University for Journalism and Communication, Bhopal, India, and an M.C.A in the year 2011 from Annamalai University, India. He qualified National Eligibility Test and was awarded Lecturership by University Grants Commission in the year 2012. Currently, he is pursuing the Ph.D program from the Department of Mathematics and Computer Science, Mizoram University. His research interests include Artificial Intelligence, Machine Learning, Deep Learning, Network Security, Cyber Security, and Machine Automated Language Translation.

Ashfaq Ahmad Najar is a Ph.D. scholar at the Central University of Kerala, India. He holds an M.Sc. in Information Technology from the Central University of Kashmir, India. He received the Young Researcher Award from the Institute of Scholars (InSc), certified under the Ministry of MSME & Corporate Affairs, Government of India, for his publication on “DDoS Attack Detection using MLP and Random Forest Algorithms.” He is currently a Graduate Student Member at IEEE, and his research interests encompass cybersecurity (DDoS Security), machine learning, deep learning, network security, and software-defined networks.

Hong Nhung-Nguyen received the B.S. degree in information technology and the master's degree in software engineering from Ha Noi National University, Ha Noi, Vietnam, in 2015 and 2018, respectively. She had pursued the Ph.D. degree in 2023 with the Information Technology Convergence Laboratory, Department of Electronic Engineering, Myongji University (MJU), South Korea, where she was advised by Prof. Yong Hwa-Kim. Since 2016, she has been a Lecturer with the Faculty of Information Technology, Viet Tri University of Industry, Vietnam. Her research interests include machine learning and software engineering.

Jamal Hussain received his M.Sc. and Ph.D. from Tezpur University (TU), Assam, India in the year 1996 and 2000 respectively. Currently he is working as Professor at Department of Mathematics and Computer Science, Mizoram University since 2007. He had conducted more than 30 International Conferences. He had guided ten Ph.D. scholars successfully and completed various project including Applicability of Artificial Neural Network for Intrusion Detection, funded by Department of Information Technology, Ministry of Communication and Information Technology, Govt. of India. His research interests include Mathematical Modelling of Biosystems, Artificial Intelligence, Deep Learning, and Network Security.

Manohar Naik Sugali is currently working as an Assistant Professor in the Department of Computer Science at the Central University of Kerala. With 10 years of teaching experience, he holds an M.C.A. and M.Tech (C.S.E) from Acharya Nagarjuna University, Guntur, India. He has been awarded a Ph.D. from the Department of Computer Science & Technology, Sri Krishnadevaraya University, Anantapuramu, Andhra Pradesh, India. He has published around 15 research papers in international journals and conferences. His current research interests include Cyber Security, Cryptography & Network Security, Intrusion Detection Systems, Pattern Matching Algorithms, Wireless Sensor Networks, and Machine Learning.