Received: 4 May 2023 | Revised: 16 June 2023 | Accepted: 14 July 2023

DOI: 10.1002/spy2.337

SURVEY PAPER

WILEY

Unleashing the power of SDN and GNN for network anomaly detection: State-of-the-art, challenges, and future directions

Archan Dhadhania[1] | **Jitendra Bhatia**[1] | **Rachana Mehta**[1] | **Sudeep Tanwar**[1] | **Ravi Sharma**[2] | **Amit Verma**[3]

[1]Institute of Technology, Nirma University, Ahmedabad, India

[2]Centre for Inter-Disciplinary Research and Innovation, University of Petroleum and Energy Studies, Dehradun, India

[3]Department of Computer Science Engineering, University Centre for Research & Development, Chandigarh University, Mohali, India

Correspondence
Jitendra Bhatia, Rachana Mehta, and Sudeep Tanwar, Institute of Technology, Nirma University, 382481, Ahmedabad, India.
Email: jitendra.bhatia@nirmauni.ac.in, rachana.mehta@nirmauni.ac.in and sudeep.tanwar@nirmauni.ac.in

Abstract

Modern computer networks' increasing complexity and scale need serious attention towards network anomaly detection. Software-defined networking (SDN) and graph neural networks (GNN) have emerged as promising approaches for anomaly detection due to their ability to capture dynamic network behavior and learn complex patterns from large-scale network data. The amalgamation of SDN and GNN for network anomaly detection presents promising opportunities for improving the accuracy and efficiency of network anomaly detection. This paper focuses on various trends, issues, and challenges by integrating GNN on the top of SDN for network anomaly detection. The article highlights the advantages of using SDN for providing fine-grained control and programmability in network monitoring. At the same time, GNN can model network behavior as a graph and learn representations from graph-structured data. The authors also discuss the limitations of traditional anomaly detection methods in SDN, such as rule-based approaches, and the potential of GNN to overcome these limitations by leveraging their ability to capture non-linear and dynamic patterns in network data. This paper also presents a case study of DoS attack detection using SDN. The result shows that SDN based approach helps to detect attacks with an accuracy of 97% with future research directions.

KEYWORDS

anomaly detection, DoS attack, graph neural networks, network security, openflow controller, software-defined networking

1 | INTRODUCTION

The rise of digital networks has increased cyber threats, and protecting these networks has become more challenging than ever.[1] Software-defined networking (SDN) is a new paradigm that offers several benefits over traditional networking, including centralized control, improved flexibility, programmability, and increased scalability.[2] In addition, SDN provides an ideal platform for implementing advanced security mechanisms, such as anomaly detection, to protect against evolving cyber threats.[3]

Anomaly detection is a critical aspect of network security. It involves abnormality identification in network traffic, finding the presence of malicious activities such as intrusion, malware, or denial-of-service attacks.[4] Anomaly detection can be challenging in complex networks with high volumes of data with the traditional approaches. The recent and upcoming technological advances in machine learning and deep learning have presented us with sophisticated anomaly detection techniques to detect and prevent threats in real-time and work on varied data and anomalies.[5] For instance, autoencoder (AE) based techniques are used for outlier reconstruction. Generative adversarial networks (GAN) and long short term memory (LSTM) approaches are promising solutions for multivariate anomaly detection.

Graph neural networks (GNN) based Deep learning approaches have proven to be beneficial in applications that involve information to be structured and represented as graphs.[5] GNNs are useful for extracting meaningful information from graph-structured data and establishing relationships among the connecting entities. It also enhances and improvises the knowledge of connecting entities through graph learning like information propagation, multi-hop learning, information aggregation, and others.[6] GNN can capture complex relationships between nodes and work efficiently in real time; this leverages the usefulness of GNN for detecting anomalies.[7] The benefits of GNN include (1) Handling complex network topology and behavior, (2) Handling real-time and dynamic anomalies, and providing quick response to potential threats. (3) Adapting to evolving network structure. These characteristics and advantages of GNN over the traditional anomaly detection approaches have led to its increased usage in the field.

This paper studies and encompasses the anomaly detection methods and assesses their ability to detect various cyber threats. The study aims to provide insights into the effectiveness of GNN for anomaly detection in SDN networks and demonstrate their potential for enhancing network security. This amalgamation will enhance anomaly detection by providing flexibility of SDN and benefits of GNN to work on complex structures, which otherwise is unavailable through traditional SDN approaches. The graph network nodes update their learning through the message-passing mechanism followed by their reliable neighbour. This model mixture can cater to various types of anomalies within the network by utilizing the notion of propagated graph learning. The significant contributions of this paper are as follows:

1. We present various approaches for detecting anomalies in SDN and provide a comprehensive review.
2. We provide the taxonomy of anomaly detection approaches traditional, SDN-based, GNN-based, and Hybrid. Their underlying principle, applicability, merits, and demerits are explored.
3. We bring various challenges and issues of anomaly detection along with the tools and technologies needed.
4. We present a case study that exercises SDN to detect DDoS attacks using the SVM model and analyze its accuracy over various types of attacks.

The rest of the paper is organized as follows: Section 2 presents a thorough background of network anomaly, SDN, & GNN. Section 3 overviews the traditional, SDN, GNN, and Hybrid approaches in anomaly detection. Section 4 discuss various recent challenges and issues. In Section 5, we explore several tools & technologies used for the development of SDN & GNN. Section 6 presents a case study of the SDN network, creates the network with the help of a mininet emulator, and analyzes the result. Finally, the paper concludes with future directions in Section 7. Figure 1 depicts the organization of the paper.

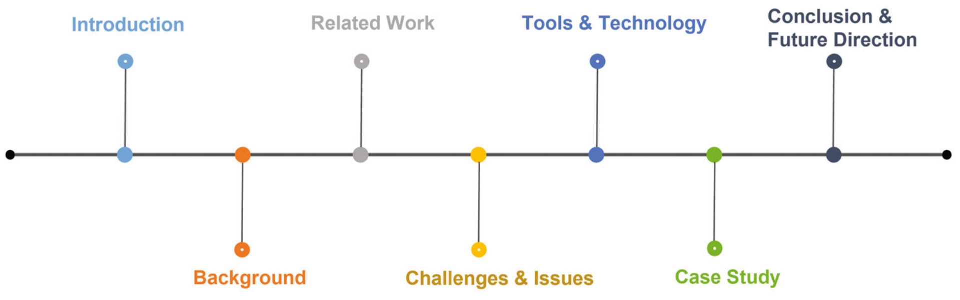

FIGURE 1 Organization of the paper.

2 | BACKGROUND

2.1 | Network anomaly

Network anomalies refer to abnormal events or patterns that occur within a network.[8,9] These anomalies can indicate the presence of security threats, performance issues, or other problems that need to be addressed. Examples of network anomalies may include sudden spikes in network traffic, unauthorized access attempts, or unusual activity from specific devices or users.[10] Detecting network anomalies is becoming increasingly important as networks become more complex and sophisticated. Detecting and responding to network anomalies, organizations can help protect their networks from security breaches and maintain optimal performance.[11] Various techniques and tools are available for detecting network anomalies, including statistical analysis, machine learning algorithms, and intrusion detection systems. These technologies can help organizations identify and respond to network anomalies in real time, minimizing the impact of security threats and improving network performance. The automatic detection and alerting of such abnormalities is the goal of anomaly detection techniques. These anomalies may indicate security lapses, malicious activity, hardware or software failures, configuration mistakes, or other unforeseen events.[12] The details of various anomalies are presented below.

1. Denial of service (DoS) attacks attempt to overwhelm a network or system with traffic or requests, resulting in degraded performance or system failure.
2. Distributed denial of service (DDoS) attacks: Similar to DoS attacks, but originating from multiple sources, making them harder to detect and mitigate.
3. Malware attacks: These are attempts to compromise a network or system by introducing malicious software, such as viruses, worms, or trojans.
4. Network scanning: An attacker probes a network for vulnerabilities or weaknesses to gain unauthorized access.
5. Insider attacks: These are attacks perpetrated by individuals with authorized access to the network or system, such as employees or contractors, who abuse their privileges for personal gain or malicious intent.
6. Data ex-filtration: This is the unauthorized transfer of data from a network or system, often for theft or espionage.
7. Network configuration errors: These are mistakes made during the setup or maintenance of a network that can result in vulnerabilities or unintended consequences.

Anomaly detection techniques Anomaly detection, also called outlier detection, is crucial in various fields such as networks, finance, cyber security, fraud detection, and healthcare. It involves identifying rare, unusual, or abnormal events or patterns that deviate from the expected behaviour in a network.[13] In this section, we will discuss five common approaches for anomaly detection: statistical-based detection, signature-based detection, machine learning-based detection, rule-based detection, and deep learning-based detection. Each technique has its strengths and limitations and can be applied in different scenarios depending on the data characteristics, domain knowledge, and specific requirements of the application. We provide an overview of anomaly detection techniques, explaining their underlying principles, how it works, and their applicability for detecting anomalies in data.[14] Figure 2 depicts the anomalies detection technique.

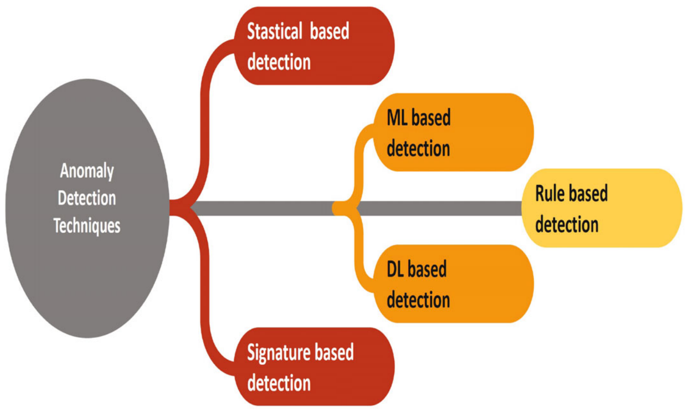
Anomaly Detection Techniques
Stastical based detection
ML based detection
Rule based detection
DL based detection
Signature based detection

FIGURE 2 Anomaly detection techniques.

1. Statistical-based detection: This approach uses statistical models to identify deviations of a network node from its expected behaviour due to network traffic. Statistical detection methods maintain the node profile indicating its features like rate of traffic, different IP addresses, connections made and several other parameters. These methods capture irregularity in node behaviour based on the variation in the network node profile. It can be analyzed using control charts, clustering, and outlier analysis. However, setting and managing the parameters of the network node profile may become challenging.
2. Signature-based detection: This method compares network traffic to a database of previously identified attack signatures. If there is a match between the current signature and the attack signature, an alarm is raised to indicate a violation. Signature-based methods have the overhead of maintaining signatures over time. The method is easy to incorporate since it involves basic signature comparison. It may fail if an unknown attack signature is encountered.
3. Machine learning-based detection: This approach tries to analyze the patterns based on the generated model. Machine learning models are adaptive and work based on the parameters fed as input and expected as output. Some algorithms used in this approach are support vector machine, decision tree, and clustering to learn from past data to identify real-time anomalies.
4. Rule-based detection: This technique defines rules or policies that describe normal behavior in a network. Any deviations from these rules can be flagged as anomalies. The rules-based detection mechanism uses an approach like sequence evaluation, an expert system based on the production environment.
5. Deep learning-based detection: This approach uses deep neural networks to learn complex patterns in network traffic and identify anomalies. This approach is suitable for high dimensional space and better efficiency than other models. Some of the frequently used techniques involve convolutional neural networks, recurrent neural networks, and autoencoders.

2.2 | Software defined network

Software-defined networking (SDN) is mainly helpful for network design and management.[15] SDN-based framework majorly comprises three planes, that is, application plane, control plane and data plane. The application plane provides the facility to interact with the underlying layer, manage, control, and change network behavior. The control plane acts as a brain to decide the routing of the data packets through the network, while the data plane is responsible for forwarding the packets.[16] The control and data planes in traditional networking are tightly coupled, making managing and scaling large networks difficult. In SDN, the control plane, is centralized and managed through software, allowing greater flexibility and programmability. SDN is based on the OpenFlow protocol.[17] The controller provides a global network view and makes intelligent decisions about route traffic based on network conditions and policies, which is especially important in today's fast-paced business environment, where organizations must adapt to changing needs.[18,19] SDN allows for network visibility and control in a better way. Since the controller provides a global network view, administrators can more easily monitor network traffic and identify potential security threats. SDN is used in various applications, including data center networking, enterprise networks, and even telecommunications networks.[20-22] It has the potential to revolutionize how we design, manage, and operate networks and is likely to become even more critical as networks grow in size and complexity.[23] These benefits make SDN preferable as compared to traditional networking models.

Figure 3 depicts the detailed architecture of SDN employing GNN. This architecture represents the integrated SDN-GNN model. The application plane first has the anomaly detection system, which locates and informs network managers of any odd or potentially dangerous behaviors within the network. This promotes network security and maintains efficient operations. Second, by offering real-time monitoring and analysis of network traffic, the network monitoring system enables managers to allocate resources better, understand network performance, and spot bottlenecks.

The control plane consists of three components, graph neural network model, analysis engine, and programmable controllers. Graph neural network uses machine learning concepts to support network optimization and decision-making by analyzing network architecture and identifying patterns. GNN helped the network for efficient learning and increased detection. The analysis engine analyses data gathered from many sources, including network devices and monitoring systems, to produce insightful conclusions and suggestions for network management. As the brain of the SDN, the programmable controller interacts with the data plane and enforces policies to manage and control network traffic flow.

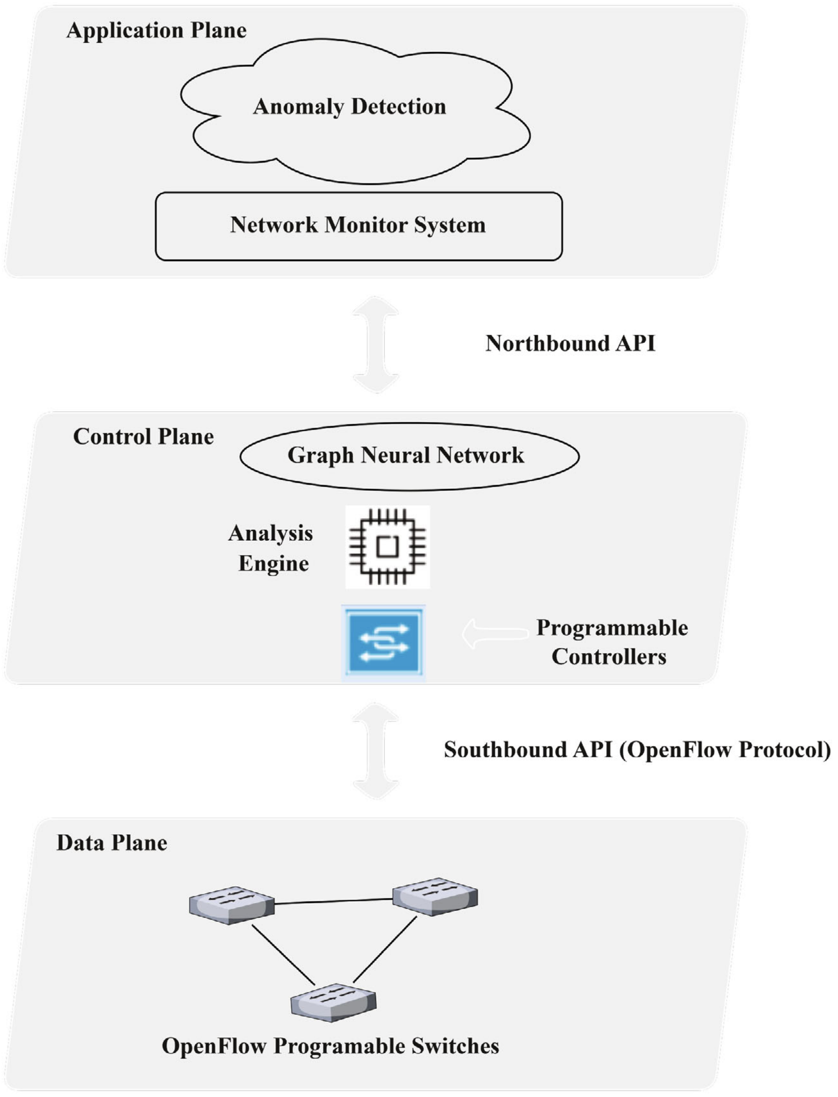

FIGURE 3 SDN controller architecture.

Network packets are forwarded and routed by the data plane, symbolized by a switch, following commands from the control plane. Making wise forwarding decisions based on the rules and policies established by the control plane ensures that data is transmitted efficiently and securely within the network.

Apart from three planes in the controller architecture, it also includes the southbound and northbound interfaces. This set of APIs is helpful in network communication. The southbound API is the SDN controller's set of interfaces for southbound communication. It allows the controller to communicate with the network devices, such as switches and routers.[24] These interfaces configure network policies and traffic forwarding rules and collect data from the network devices, such as traffic statistics and network topology. The southbound API is responsible for implementing the OpenFlow protocol, SDN's most commonly used protocol. It enables the SDN controller to manage the forwarding of packets within the network by programming the flow tables of the switches.

The northbound API is used by the SDN controller for northbound communication, which allows external applications or network management systems to communicate with the SDN controller. For example, an application could use the northbound API to request a network policy change or to retrieve network topology information. In addition, the northbound API enables integration between the SDN controller and other systems, such as orchestration platforms, network management systems, and cloud platforms. This integration allows for greater automation and orchestration of network functions and services. It enables the controller to be used in various applications, including data center networking, enterprise networking, and telecommunications networking. SDN controllers and network devices communicate via the southbound API. In contrast, the northbound API connects the SDN controller to other applications or network management systems.

Like any other network, SDN networks are vulnerable to various attacks. Here are some of the possible attacks on SDN networks:

1. Denial of service (DoS) attacks: SDN controllers can be overwhelmed with a large volume of requests, making them unable to respond to legitimate requests from network devices and applications. This can cause network disruptions and downtime.
2. Man-in-the-middle (MitM) attacks: An attacker can intercept and manipulate the communication between the SDN controller and network devices, compromising the security and integrity of the network.
3. Switch flooding attacks: An attacker can flood the network with many bogus flow table entries, causing switches to become unresponsive and disrupting network traffic.
4. Control plane attacks: An attacker can target the control plane of the SDN network, compromising the controller and its ability to manage and control the network.
5. Malware attacks: SDN networks can be infected with malware, such as viruses, worms, and Trojans, compromising network security and integrity.
6. Spoofing attacks: An attacker can spoof the identity of a network device, such as a switch or a controller, to gain access to the network and launch further attacks.

2.3 | Graph neural network

Graph neural networks (GNNs) are machine learning algorithms well-suited for analyzing graph-structured data, such as social networks, molecular structures, and communication networks. For example, in anomaly detection, GNN can be used to learn the expected behavior of a system and then identify anomalies or outliers based on deviations from that behavior.[25] In an SDN-based solution, GNN can analyze network traffic flow and detect anomalies such as network intrusion, DDoS attacks, or abnormal traffic patterns.[26] SDN is an architecture that allows network administrators to programmatically control and manage network behavior programmatically, making deploying and managing complex networks easier. Using GNN in an SDN-based solution, network administrators can analyze traffic flow through the network and detect anomalies in real-time.[27]

One common approach to using GNN for anomaly detection in SDN-based solutions is representing the network as a graph. Each node represents a device or network component, and each edge represents a connection between those components. The GNN can then be trained to learn the expected behavior of the network by analyzing patterns in the traffic flow and the interactions between network components. Once the GNN has understood what constitutes normal behavior, it can be used to identify deviations from that behavior and flag them as potential anomalies. Another approach uses GNN to analyze the traffic flows rather than the network structure. In this approach, traffic flows are represented as graphs, where each node represents a packet, and each edge represents the flow of that packet through the network. The GNN can then be trained to learn the standard traffic flow patterns and identify deviations from those patterns. GNN computational model is built with the help of the propagation, sampling, and pooling models. The typical GNN architecture is in the middle part. The pooling module is applied to extract high-level information after the convolutional operator, recurrent operator, sampling module, and skip connection are utilized to propagate information in each layer. Figure 4 presents the generalized architecture of GNN.

The parameter and model selection in graph neural network is dependent on the following factors:

Node features: Let $X \in \mathbb{R}^{N \times D}$ represent the node feature matrix, where N is the number of nodes in the graph and D is the dimensionality of the node features. This feature represents the current node concerning the features considered.

Edge features: Let $E \in \mathbb{R}^{N \times N \times M}$ denote the edge feature tensor. Here M represents the number of edge features, and $N * N$ is the pair of nodes connected to the given edge.

Graph structure: The graph structure can be represented by the adjacency matrix $A \in \{0, 1\}^{N \times N}$, where $A_{ij} = 1$ if there is an edge from node i to node j, and $A_{ij} = 0$ otherwise.

Node updation & aggregation: This is an important part of GNN Node learning. At first, information aggregation from the neighbouring node takes place, and then aggregated information is used to update the current node's features. It involves the activation function, denoted by $f(\cdot)$, which helps in information propagation and aggregation.

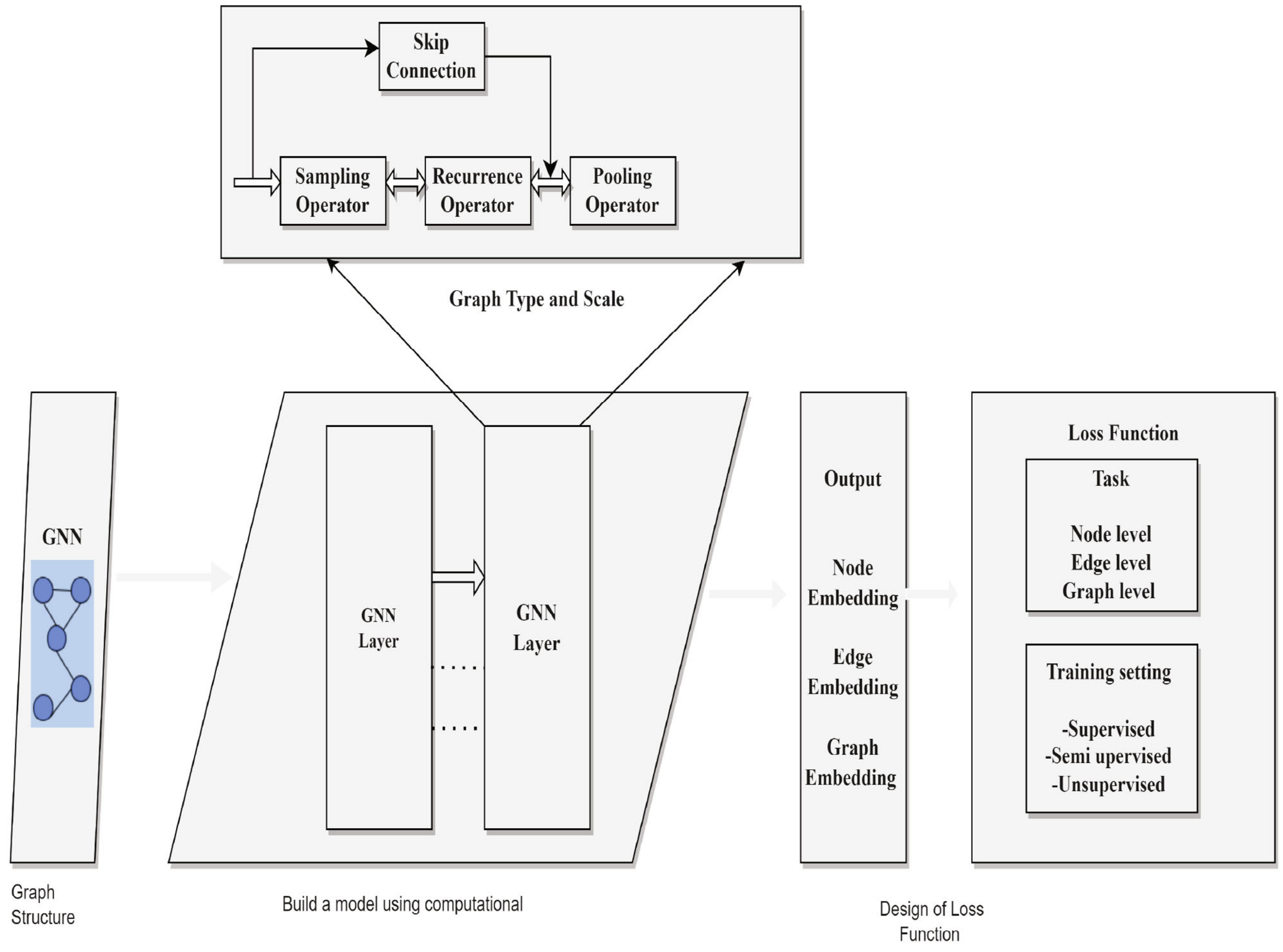

FIGURE 4 GNN architecture.

Each individual operation can be defined as $H^{(l+1)} = f(H^{(l)}, A, E)$, where $H^{(l)}$ represents the node representations at that layer l.

Hyperparameters: Let θ denote the set of hyperparameters, including the learning rate, number of layers, activation functions, regularization terms, and dropout probability. These parameters are tuned during training to optimize the GNN's performance.

The integration of SDN and GNN helps to withstand the following attacks:

1. Distributed denial of service attacks: Real-time detection and mitigation of DDoS assaults are made possible by SDN's centralised control and network traffic visibility. Adding the GNN model on the control plane will help to detect anomaly traffic and provide necessary countermeasures through insight into traffic patterns.
2. Malware propagation: Network managers can impose stringent security standards and divide the network to slow the spread of malware thanks to SDN's centralised control. To enable rapid reaction and containment, GNN-based algorithms can help identify and evaluate network traffic patterns linked to malware activities.
3. Insider threats: SDN enables fine-grained access control and network traffic monitoring, making it simpler to spot suspicious activities or unauthorised insider access. GNN models can learn from past data and spot irregularities in user activity, allowing for the early identification of potential insider threats.
4. Zero-day exploits: To quickly install security fixes and isolate the impacted components, SDN can dynamically rearrange the network. GNN models can help identify network traffic patterns linked to zero-day attacks, allowing preventative action to be done to stop their exploitation.

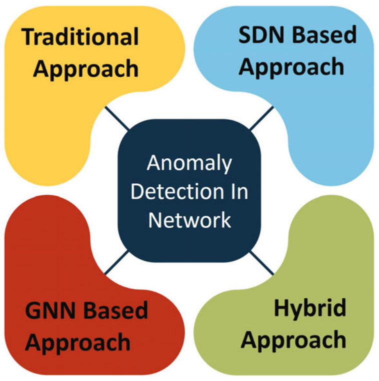

FIGURE 5 Taxonomy.

3 | RELATED WORK

Network anomalies, such as cyber-attacks or system failures, can have serious consequences, such as data loss or network downtime. Therefore, detecting anomalies in a network is an important task. Network anomaly detection is one of the research areas investigating the combination of SDN and GNN. Several studies have proposed using GNN and SDN to detect network anomalies. For example, some researchers have proposed using GNN to learn the expected behavior of a network and then use this knowledge to identify deviations from this behavior as potential anomalies. Other work has focused on using GNN to analyze real-time network traffic and flag suspicious activity. Using GNN in combination with SDN for network anomaly detection can improve the accuracy and efficiency of anomaly detection and help protect networks from various threats. Figure 5 depicts the taxonomy of the paper's literature survey.

3.1 | Traditional approaches

The traditional anomaly detection approach uses standard machine learning techniques for anomaly detection. In addition, these machine-learning approaches incorporate supervised and unsupervised learning techniques. The methods and algorithms used to detect anomalies in the traditional approach are explained and mentioned in the Table 1. In Reference 28, the authors proposed a method that gives the information to evaluate the performance of two machine learning approaches: supervised and unsupervised. This paper mainly focuses on the cyber security domain and intrusion detection in the cloud, IoT and SDN. KDD Cup'99 & DARPA 1999 dataset used for the experiment. In Reference 29, authors have proposed supervised, unsupervised, and semi-supervised learning algorithms for intrusion detection in IoT networks. For this purpose, the KDDCUP99 dataset and ISCX dataset are used. Finally, in Reference 30, the authors proposed a method focused on malware analysis in Windows, giving the results about intrusion and anomaly detection. They used supervised, unsupervised, and semi-supervised learning approaches for anomaly detection.

In Reference 31, the authors propose a method of various machine learning approaches to detect anomalies in cybersecurity. The author discusses supervised and unsupervised algorithms, reinforcement learning, and anomaly detection using feature selection methods in this work. It also discusses identifying potential malicious devices using the probabilistic generative adversarial networks (Pgans) model. In Reference 32, the authors proposed a method of intrusion and anomaly detection is a vital issue in IoT security. This work shows that use case machine learning approaches like supervised, unsupervised, and semi-supervised algorithms help discover intrusion and anomalies in IoT Security. Finally, in Reference 33, the authors of the proposed method use different machine learning approaches for malware detection and categorization using unsupervised and supervised learning in the Windows operating system.

3.2 | SDN based approaches

SDN-based anomaly detection involves monitoring the network traffic flow and identifying abnormal patterns or behaviors that deviate from the usual network behavior. SDN controllers can dynamically reconfigure the network topology

TABLE 1 Traditional approaches.

References	Objective	ML models	Target domain	Research limitations
Hodo et al.[28]	Classify the existing IDS approaches	Learning that is supervised & unsupervised	Detecting intrusions in the cloud, IoT, and SDN	Doesn't provide the technicality
da Costa et al.[29]	ML based IDS	Learning that is supervised, unsupervised, & semi-supervised	IoT network intrusion detection	Lack of real world validation
Ucci et al.[30]	Malware analysis	Learning can be supervised, unsupervised, or semi-supervised	Analysis of malware in Windows	Lack of depth malware analysis
Tahsien et al.[31]	ML based IoT security solution	Learning that is supervised, unsupervised, & reinforced	IoT security	Doesn't provide the accurate detail in IoT
Hussain et al.[32]	Evalute the exiting ML based IoT security	Learning that is supervised, unsupervised, semi-supervised, & reinforced	IoT security	Need of dataset can't test on the real time data
Gibert et al.[33]	ML based malware detection	Learning that is supervised & unsupervised	Windows malware identification & categorization	Can't measure the adversarialattack

and route the traffic in real-time to mitigate any detected anomalies. It leverages flexibility, scalability, and automation compared to traditional anomaly detection techniques. The SDN-based approach summary is given in Table 2.

In Reference 34, the authors proposed a method to detect MitM attacks and deploy a supervised learning (SL) model in the southbound interface (SBI). If a node is abnormal, the model seeks to disconnect it. The state of a network node, time to live, and response time are indicators of suspicious requests, and the data gathered by SBI is classified as normal or abnormal based on these factors. The random forest (RF) algorithm uses the labelled data to train a classifier that approves or rejects new connection requests. In Reference 35, the authors proposed method SDN has become an essential topic of research and development because of its potential to create more secure networks by decoupling the control plane and data plane. Implementing SL in SDN can significantly reduce the attack surface by removing one line of potential vulnerability. This paper describes another promising method based on time-sensitive routing (TSR) that can best use time windows, namely the small window size. It refers to a brief period before a decision is made, which can result in early detection or a reduction in accuracy if there is insufficient data to make the correct prediction.

In Reference 36, the authors proposed a method to minimize damage caused by attacks on the control plane of an SDN network. This work focuses on developing control plane security checks to detect suspicious transactions using distributed security model system (DSMS). In this model, centralized SDN monitoring facilitates the security model to determine if the request is malicious by periodically collecting statistics from the switches.[37] The predictor is a trained-to-model, and based on the output, inputs are fed to relay devices to forward or drop packets. In Reference 38, the authors proposed method explores three categories of DDoS attack targets in an SDN network that can be affected the controller, flow, and bandwidth tables between a switch and a controller. The test results show that the most important characteristics for accurate detection are the IP source port and the number of packets and bytes in the stream. However, the machine learning model's performance in detecting anomalies may need to be more accurate, so an additional step, such as entropy measurement, can improve its performance.

In Reference 39, the authors proposed an approach that collects statistics from the network at regular intervals and calculates the entropy. If the entropy value exceeds a predefined threshold, an SL classifier determines whether the network is under attack. This two-step approach can improve attack detection accuracy by considering the network traffic's complexity and randomness. In Reference 4, the authors proposed a method for predicting threats in an SDN network

TABLE 2 SDN based approaches.

References	ML model	Anomaly	Methodology	Advantages	Research limitations
Sebbar et al.[34]	Random forest	MitM	Classify connection requests made through SBI in SDN using an SL model.	-Time complexity decreased - Malicious node detection is expedited.	An environment under control is used for the test.
Khamaiseh et al.[35]	SVM, NB, KNN	Saturation attacks	Train an SL model via the time-window of traffic.	-Identify unknown Saturation attacks.	The configuration of the test environment affects the detection performance.
Akbas et al.[36]	SVM, NB DT, and KNN	Misbehavior attacks, flow attacks and DoS/DDoS attacks	Get the traffic data via controller and train an SL model offline	- Easy to collect real time data - Deploy ML models	The datasets that were utilized for validation are out of date.
Santos et al.[38]	SVM, DT, RF, multiple layer perception (MLP)	DDoS attack	Check the effectiveness of SL models for spotting DDoS attacks.	- SDN properties important for DDoS detection.	There is no discussion of recall or precision.
Banitalebi Dehkordi et al.[39]	BayesNet, DT, random tree, logistic regrassion	DDoS attack	Find suspect data using entropy, and train an SL model to perform additional verification.	- Two-step check boosts precision.	The tenfold categorization method's use of resources is unmentioned.
Wang et al.[4]	RF	DoS attacks and etc	Data prediction train SL model offline and if any quarry is their entropy value is major.	- Decreases uncertainty when judgments are made with a limited feature set.	The dataset that was utilized for validation is out of date.
Simpson et al.[40]	Semi gradiant sarsa	DDoS attacks	Through source-destination pair analysis, teach an RL model to drop packets.	- The mitigation strategy permits traffic to pass through while assaults are taking place - The RL model acts per flow.	No specific performance information is given.
Sampaio et al.[41]	Q-learning	Flooding attacks	Balance the load over all SDN links, use an RL model.	- Without human involvement, advanced route management.	More switches joining the network will cause the number of states to grow exponentially.
Han et al.[42]	Double deep Q-networks and asynchronous advantages actor-critic	Cyber attacks aiming at compromise network nodes.	Prevent network nodes from being compromised via the node and link state, use an RL model. Adversarial training should be used to safeguard the RL model.	- SDN provides autonomous defence, which lessens attacks on the RL model itself.	It is predicated on the idea that the attacker has access to every node along the transmission channel.

using three subsystems that run on the SDN controller. The first subsystem filters and processes the data to extract critical information and discard irrelevant data, providing valuable clues for the RF classifier. Based on the predictions from the RF classifier, regular requests are processed, and abnormal data are blocked. When the classifier's decision is ambiguous, the system uses entropy measurement to assess the level of ambiguity and make a final decision. In Reference 40, the authors proposed a method for dropping packets in a switched packet network using two different modes of operation. An agent instantly chooses the probability of dropping packets to partially discard existing traffic streams while allowing at least 10 flows to pass through. This mode does not take into account the future state of the network. Protected mode blocks traffic and modifies agent state action values based on network state. Evaluation results reveal that instant agent mode works well in a multi-agent environment where multiple agents guard traffic to the same server separately.

In Reference 41, the authors proposed a model that uses reinforcement learning (RL) to detect malicious traffic in an SDN network. The model uses RL to balance load by monitoring the links and modifying the routing when a link has a load of over 80%. The SDN controller sends a signal to the RL agent to update the route, and a positive reward is given if there are no high-status links after the update. Additionally, harmful traffic can be redirected using this technique. In Reference 42, the authors propose a new RL architecture that uses adversarial training to mitigate the impact of RL during training. Each node and link in the network has two possible states: normal or compromised and on or off, respectively. The network comprises nodes and links, and its current state can be seen from the combination of node and link states. Nodes or links can be switched on or off, or nothing can be done, depending on the condition. In RL-based mechanisms, influencing factors such as essential server availability, cost of mitigation, and the number of accessible network resources can be considered for the reward. An adversary agent can adversely impact the attacker by producing adversarial outputs because the attacker is aware of the agents and can imitate their results, thereby protecting the RL from malicious attacks.

3.3 | GNN based approaches

Graph neural networks (GNNs) are a type of neural network designed to process graph-structured data. Anomaly detection using GNN involves training a GNN on a graph dataset, where anomalies are labelled as "outliers" or "abnormal." The trained GNN can then predict anomalies in new, unseen graph data. GNN-based anomaly detection has shown promising results in various domains, including cyber-security and social network analysis. In Reference 43, the authors proposed a model with convolutional neural networks, long short-term memory (LSTM), and deep neural networks used to detect an anomaly as the primary findings of this paper. The used dataset is Webshop S5 provides a better and more accurate result with CNN & LSTM technology. However, this model needs to find the optimal parameter. In Reference 44, the authors proposed an approach for this model to use a real-time data set, which provides an accurate result and also presented the heterogeneous anomaly detection in social diffusion (HADISD) framework that utilizes the provided data to create performance prediction models.

In Reference 45, the authors proposed a model that will show an LSTM-based model that is applied to the DBLP dataset and give some comparison between the results from this model and others. In Reference 46, the authors proposed a fuzzy model approach Neuro-fuzzy horizontal anomaly detection (NHAD) model proposed here for the used dataset is DAPRA98. The model is occasionally effective, and it is susceptible to failure. It could be solved by increasing the diversity of terms or by adding symmetrical words. In Reference 47, the authors proposed a graph convolution network (GCN) method that used a technique and dataset MNIST. This model and dataset provide good results in a non-euclidean domain. The summary of GNN based approach is given in Table 3.

3.4 | Hybrid approaches

This section represents the work that considered the amalgamation of SDN and GNN for detecting various anomalies. These hybrid approaches are efficient as compared to other models. The summary of the hybrid approach is given in Table 4. In Reference 48, the authors proposed an approach MLH-IDS hybrid technique that combines supervised and unsupervised learning. It has three stages; The first level employs supervised machine learning CatSub+ to classify DoS and Probe, the second level employs unsupervised machine learning K-point method to detect normal traffic, and the third level employs outlier-based classifier GBBK to classify R2L and U2R. All types of attacks were effectively detected

TABLE 3 GNN based approaches.

References	Technique	Results	Dataset	Advantages	Research limitations
Kim and Cho[43]	CNN + LSTM + DNN	Accuracy: 98.60%	WebScop S5	Provide better and accurate result	Can't find optimal parameter
Liu et al.[44]	Parameter–free Framwork (HADISD)	Precision: 89.7%, Recall: 91.3%, FI-score: 90.5%	Real-life datasets	Efficient and effective proposed schemas	For other schemas that can't perform well
Aggrawal and Arora[45]	LSTM	Accuracy: 84%	DBLP	Result is good for particular dataset	More specific not give good result for other algorithms
Sharma et al.[46]	A self healing neuro-fuzzy approaches (NHAD.)	Accuracy: 99.98%, Precision: 98.1%, DetectionRate: 97.97%	DAPRA 98	Efficient model	Sometime model fails
Monti et al.[47]	GNN (graph neural network)	Accuracy: 81.50%	MNIST	Good for non-euclidian domain	For other not provide batter result

with the assistance of NSL-KDD and a Multilevel hybrid intrusion detection system (MLH-IDS). However, the actual performance of the system is still unclear because the authors identified the assaults that are present in KDDTest+ but not in KDDTrain+ as unknown during the testing phase. In Reference 49, the authors proposed approaches to give a detailed analysis of an artificial intelligence application used to detect unknown attacks during the training process. The author focused on the grey area and distributed-data concept. Different ANN parameter sets were assessed using a two-tier classifier with LDA feature selection.

In Reference 50, the authors proposed a method It was shown how to combine two-step binary classification using decision trees, random forests, and KNN classifiers as the basic classifiers. Experiments were performed on different feature sets in different attack patterns to select the best classifiers. But when employed on probe attacks, the results showed a high false alert rate and subpar performance. As such, another experiment focused on adapting the accuracy while keeping a low false alarm rate. Here, even a much more aggressive classification method (e.g., a deep neural network) resulted in good accuracy despite the lack of feature set similarity between training and test data samples, which we would expect to be needed by DNN for learning from training data sets. In Reference 51, the authors proposed the hybrid multilevel data mining framework method to detect and prevent advanced threats in networked systems, proving that mixed feature selection outperforms traditional distance-based selection, especially in large datasets. Therefore, the authors suggested a hybrid feature selection-based multilevel data mining approach. They conducted numerous experiments with various ML methods to select the best algorithms for identifying each attack, leading to outstanding results. However, this approach has four classifiers, which could be too much for real-time IDS.

In Reference 52, the authors proposed method presents a hybrid IDS in this work that integrates various feature selection strategies. The suggested IDS has four different methodologies for each attack class, including RF to detect DoS, the Stacking method with RF, J48, and KNN to detect R2L, to be constructed multilevel manner. Then, it produces traffic from these data by utilizing decision trees and the cfsSubsetEval and wrapperSubsetEval feature selection algorithms on the various feature selection techniques used in the proposed IDS. In Reference 53, the authors proposed an approach to handle the training set's skewed class distribution and devised the MultiTree method. To lessen bias against overrepresented classes, it modified a piece of the training data set. To choose the basic classifiers, the authors tested a variety of classifiers, including deep neural networks, decision trees, random forests, and KNN. The ultimate prediction was made via adaptive majority voting.

In Reference 54, the authors proposed method has multiple parameters in today's network assaults for ML to learn. As a result, feature selection is a crucial step in IDS. Recent studies have shown that many researchers investigate the best feature selection techniques to extract a subset of pertinent features from data sets to improve classification results. Examples of these techniques include using the local search algorithm with K-means or particle swarm optimization (PSO). Furthermore, artificial neural networks (ANN) and deep learning (DL) have been successfully used in recent years to cope with complicated patterns, particularly in processing images and languages. In Reference 55,

TABLE 4 Hybrid approaches.

References	Main contribution	Proposed methods	Algorithms	Feature selections	Anomaly detection	Evalution criteria	Dataset
Gogoi et al.[48]	Three-level hybrid IDS using supervised, unsupervised, and outlier detection.	MLH-IDS	CatSub+, k-point, and GBBK	Mutual information	DDoS	Detection rate FDR	NSL-KDD Cup 1999
Pajouh et al.[49]	Two-layer dimension reduction and two-tier classification	TDTC	NBC and CF-KNN	PCA and LDA	User to root (U2R) and remote to local (R2L)	Detection rate FAR	NSL-KDD
Li et al.[50]	An effective two-step hybrid using binary classification and KNN	Effective two-step IDS	C4.5 DT, KNN	Gain ratio in C4.5	U2R & R2L	Accuracy, F1 score, Precision detection rate FAR	NSL-KDD
Yao et al.[51]	HMLD using hybrid feature selection and hybrid classification	HMLD	SVM, ANN	FMIFS	DoS & R2L	Accuracy, F1 score, Precision detection rate	KDDCUP99
Cavusoglu[52]	Hybrid-layered IDS using four different classifiers to detect each attack type	Hybrid-layered IDS	RF, J48 DT, KNN	CfsSubsetEval and wrapper-subsetEval	DoS, R2L & U2R	Accuracy, F1 score, detection rate TPR FAR MCC	NSL-KDD
Gao et al.[53]	Adaptive ensemble using multiple ML algorithms, and adaptive voting algorithm	Adaptive ensemble	DI, RF, KNN, DNN, and multitree	CART	DoS, R2L & U2L	Accuracy, F1 score, precision detection rate	NSL-KDD
Tama et al.[54]	Two-stage meta classifier with majority voting and hybrid feature selection	TSE-IDS	Rotating forest and bagging	PSO+ACO+GA	DoS & DDoS	Accuracy, precision detection rate FAR	USNW-NB15
Golrang et al.[55]	Hybrid multi-objective approach to address the redundant feature selection issue	Hybrid and multi objective approaches	Random forest	NSGAII-ANN	DoS	Accuracy, F1 score, Precision detection rate FAR	KDDCup99
Liu et al.[56]	Hybrid IDS using scalable K-means random forest, and CNN+LSTM for anomaly classification	Hybrid K-mean + RF	K-means, RF, and CNN+LSTM	Attribute ratio	DoS & DDoS	Accuracy, TPR	NSL-KDD

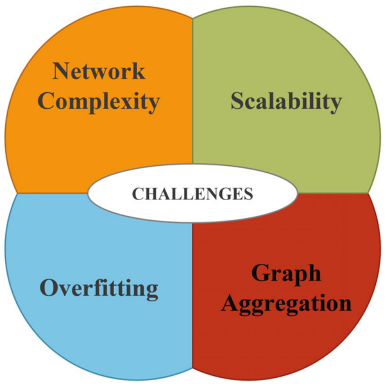

FIGURE 6 Challenges.

the authors proposed method is a Hybrid and multi-objective approach to studying the detection of redundant features in textual data sets is proposed in this paper. The use of data sets from KDDcup99 is discussed as an example. Random forest algorithms are used to prevent DoS-type attacks on the control layers for this user data set. In Reference 56, the authors proposed a hybrid IDS that classify anomalies using scalable K-means random forest and CNN+LSTM. DoS and DDoS attacks of the control layer are employed in this approach to target the control layer for the NSL-KDD data set.

4 | CHALLENGES & ISSUES

In today's IT industry, the scenario for SDN is highly favorable. However, deploying SDN and utilizing it in real-time networking poses specific challenges. GNN is a deep learning-based approach that has gained popularity and is also used in real-time use cases. However, GNN also faces some challenges, which are discussed here. Furthermore, these challenges are described in this context when we combine both technologies. Figure 6 shows the enlisted challenges for the existing SDN and GNN anomaly detection systems.

4.1 | Challenges

1. Network complexity: While SDN promises increased network programmability and automation, it can also introduce complexity in network management. SDN environments may require new skills, expertise, and tools to configure, monitor, and troubleshoot the network. Training network administrators and staff on SDN concepts and technologies and implementing practical management tools can help mitigate this challenge.
2. Scalability: SDN networks can grow in size and complexity quickly, resulting in challenges related to scalability. As the number of network devices and flows increases, the network controller may struggle to handle the increased load, resulting in performance issues and potential network congestion. Proper network design and controller scalability are crucial to ensure that SDN networks can handle the growing demands of modern networks.
3. Overfitting: Deep learning models such as GNN can also suffer from overfitting, where the model may overly memorize the training data and struggle to generalize to unseen data. Graph data can have unique challenges in terms of overfitting, such as the presence of highly connected nodes or small subgraphs. Developing effective regularization techniques and strategies for generalization in GNN is an ongoing research topic.
4. Graph aggregation: Representing graphs in a way neural networks can process is a fundamental challenge in GNN. Graphs can have varying sizes, structures, and connectivity patterns, which require effective aggregation strategies to combine information from neighbouring nodes or edges. Designing effective and scalable graph representation and aggregation methods is critical to the performance of GNN.

4.2 | Drawbacks of integrated model

Integrated model of SDN and GNN is very efficient in anomaly detection but still it has some drawbacks which are shown in Figure 7.

1. Integration: Integrating SDN and GNN in existing network environments can be challenging. SDN may require changes to the network infrastructure, such as deploying OpenFlow switches, while GNN may require modifications to the data processing pipeline and network management systems. Coordinating the deployment and integration of SDN and GNN can be complex, as it requires coordination between network operators, data scientists, and system administrators. Developing effective strategies for deploying and integrating SDN and GNN in real-world network environments is essential for practical adoption.
2. Performance evaluation: Evaluating the performance of a combined SDN-GNN system can be challenging, as it may require considering both network-level performance metrics, such as latency, throughput, and network utilization, as well as application-level performance metrics, such as accuracy, robustness, and efficiency of the GNN model. Therefore, developing appropriate evaluation methodologies and metrics for a combined SDN-GNN system is essential to ensure its effectiveness in real-world applications.
3. Computational complexity: Particularly for massive networks with millions of nodes and edges, GNNs can require significant processing resources. The overall computing complexity of network operations and decision-making processes may grow due to integrating GNNs with SDN. This might not be preferable in some network conditions because it can result in higher latency and decreased real-time responsiveness.
4. Scalability challenges: SDN architectures are made to handle huge networks and scale well. However, because GNNs sometimes involve intricate computations on graph structures, integrating them could provide scalability problems. The computational demands of GNNs may create a bottleneck as the network size increases, restricting the system's ability to scale.
5. Training and learning overhead: To capture the graph structure and discover the underlying patterns, GNNs often need training and learning methods. Training GNN models on network data, which can be time-consuming and computationally costly, is a prerequisite for integrating GNNs into an SDN environment. Additionally, constant learning and adapting GNN models could result in added overhead as the network develops.
6. Data requirements and privacy concerns: To efficiently carry out their computations, GNNs depend on graph data. Beyond what is typically used in SDN systems, more network data may need to be gathered and processed to integrate GNNs with SDN. When sensitive information is present in the network data, this may give rise to privacy problems. When merging SDN with GNN technology, it is vital to ensure data privacy and protection.
7. Model interpretability: GNNs are renowned for being "black boxes", which makes it difficult to understand how the models make decisions. The inability of GNN models to be interpreted can be a disadvantage in SDN systems where

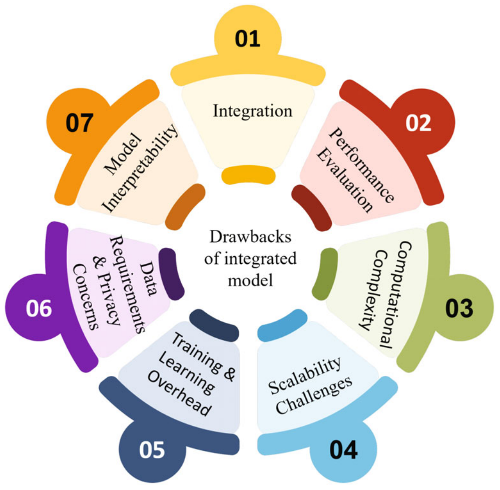

FIGURE 7 Drawback of involving SDN & GNN model.

TABLE 5 Tools & technology.

Tool	Programming language	Features	Limitations
POX	Python	Highly customizable, easy to use	Lacks of load balancing & network virtualization
NOX	Python or C++	Highly customizable	Lack of network virtulization
OpenDayLight	JAVA	Highly scalable, web based interface	Difficult to set up and configure
FloodLight	JAVA	Highly modular & customizable, web based interface	Complex to set up & configure, lack of network virtulization
RYU	Python	Highly customizable, support multiple protocols	Difficult to set up & configure
PyTorch geometric	Python, PyTorch	Provide wide range of tools & functionality	Difficult for beginner
Deep graph library (DGL)	Python, PyTorch, TF (TensorFlow), MxNet	Supports multiple backends, provide the wide range of tools and functionality	Difficult for beginner, also need more memory & more processing power
Graph nets	Python, PyTorch	Easy to use, simple interface	Not for larger- scale applications
Spektral	Python, TF2/Keras	Powerful functions and tool UI is friendly interface	Not suitable for beginners

network administrators must comprehend and approve network decisions. Combining SDN and GNNs can present a substantial difficulty in terms of ensuring transparency and interpretability in the decision-making process.

5 | TOOLS & TECHNOLOGY

Several tools and technologies are available for implementing SDN and GNN, including POX, NOX, Ryu, OpenDaylight, Floodlight, PyTorch Geometric, Deep Graph Library (DGL), Graph Nets, and Spektral. POX and NOX are open-source SDN controllers that provide a Python-based programming interface, while Ryu is a popular SDN controller written in Python with a plugin architecture. OpenDaylight is a modular SDN Java-based platform that supports multiple south-bound protocols. Floodlight is an open-source SDN controller written in Java and known for its high performance and scalability. PyTorch Geometric and DGL are libraries for GNN development, with PyTorch Geometric being built on top of PyTorch and DGL, providing flexible implementations for large-scale graph processing. Finally, graph Net is a library for building graph networks, and Spektral is a Keras-based library for GNN development that provides high-level APIs for constructing GNN models.[57] A summary of various tools & technologies is presented in the Table 5.

6 | CASE STUDY

In modern communication systems, networks connect businesses, organizations, and individuals. However, network anomalies such as cyber-attacks, congestion, and failures can disrupt operations and compromise network security. Thus, network administrators and security professionals must detect and respond quickly to anomalous network behavior well in advance. Common attacks on network infrastructures include DoS, switch flooding, malware attacks, and spoofing attacks. SDN is appropriate to address such challenges as it provides a centralized control plane that enables network administrators to identify and respond to network anomalies quickly. It is crucial to promptly detect and address malicious networks or machines to ensure they are malware-free. Various machine learning algorithms can be applied to achieve more accurate results in detecting and responding to network anomalies. These include support vector machines, random forests, K-nearest neighbors, decision trees, reinforcement learning, and other deep learning (DL) based algorithms like artificial neural networks, graph neural networks, graph convolution networks, and convolution neural networks. These

TABLE 6 Experimental setup.

Parameter	Value
Platform	Intel Core i3, 8 Gb RAM
Operating system	Ubuntu 20.04 LTS
Type of traffic	TCP, UDP, ICMP
Traffic generator	Hping3
Type of attack	DoS
Packet used	200, 600, 1000
Controller	Floodlight
Switch	OpenFlow switch
Network emulator	Mininet

ML and DL algorithms can be applied to the SDN control plane, enabling administrators to make more informed network operations and security decisions.

Table 6 gives a brief idea about the experimental scenario carried out for this case study. This study analyzes the flow status information. The analysis aims to extract characteristic values associated with DoS attacks. To differentiate normal traffic from abnormal traffic that indicates an attack, an SVM-based algorithm was used to classify the characteristic values information.[58] Several parameters were considered to define the accuracy and efficiency of the network. Those parameters are defined next part below.

(1) The rated speed of source IP (SSIP) is defined as the number of source IP addresses generated within a specific unit of time.

$$\mathrm{SSIP} = \frac{\mathrm{Total\_count\_of\_IP}}{\mathrm{time}} \tag{1}$$

The variable addition of total IP represents the total number of source IP addresses observed within a given sampling interval and time. For example, during an attack, attackers may generate many data packets with randomly forged source IP addresses, rapidly increasing the observed number of source IP addresses.

(2) The source port speed (SSP) refers to the rate at which source ports are generated or utilized within a given time frame.

$$\mathrm{SSP} = \frac{\mathrm{Total\_Port}}{\mathrm{time}} \tag{2}$$

When there is a high volume of attack requests, many port numbers are generated randomly, resulting in an increased count of attack source ports, denoted as "total ports".

(3) The rate of flow entries (RFE) indicates the number of flow entries per unit time.

$$\mathrm{RFE} = \frac{\mathrm{Number\_of\_packets}}{\mathrm{time}} \tag{3}$$

During an attack, the number of flow entries per unit of time increases significantly, surpassing the normal value by a significant margin. Therefore, detecting a DoS attack can be viewed as a two-classification problem. To accomplish this, we utilize the SVM algorithm and gather switch data to extract feature values for training. We aim to identify the optimal classification hyperplane that separates normal data from DoS attack data. Subsequently, we evaluate our model using test data to obtain classification results. The network topology for the experiment is generated using the Floodlight controller and OpenFlow switch, deployed on the Ubuntu operating system. The topology is created using Mininet. The detection method for DDoS attacks is validated by deploying an SDN environment. H1 and H2 are used as hosts, while H5 serves as the victim target. Normal and DoS attack packets can be sent from H1 and H2 to generate corresponding samples. H3 and H4 are used to create normal network traffic samples for training the model and detecting attacks.

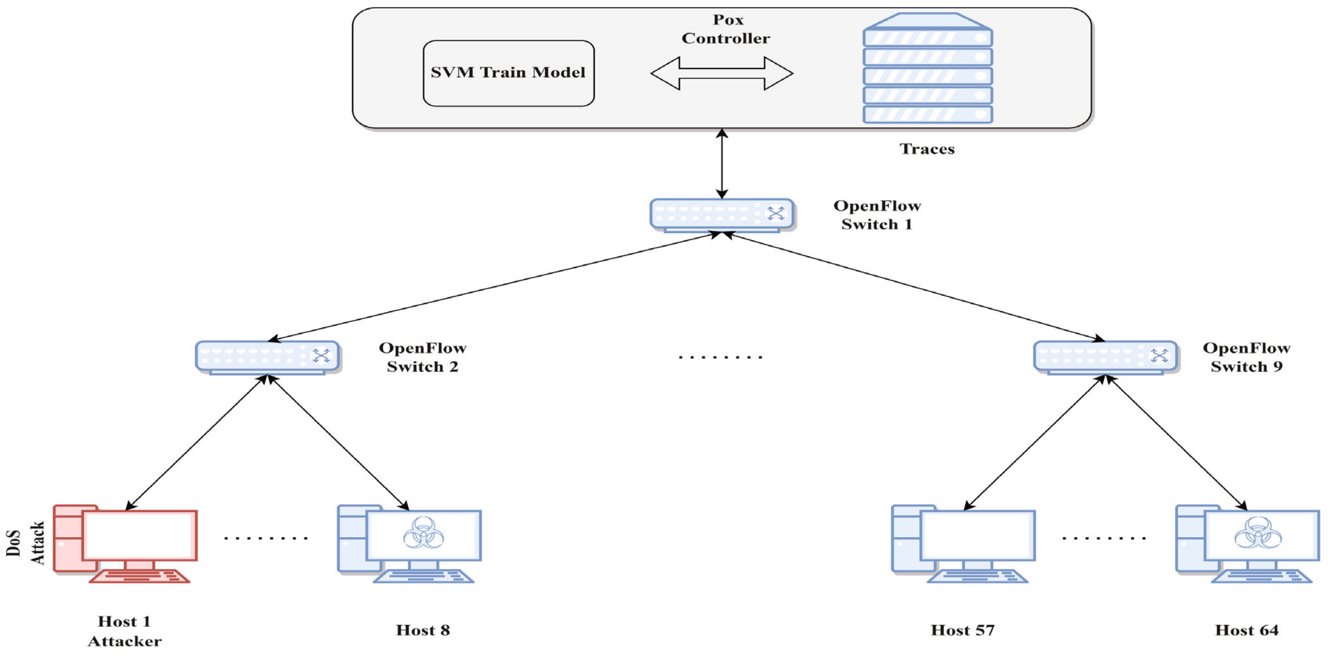

FIGURE 8 SDN based DDoS attack detection.

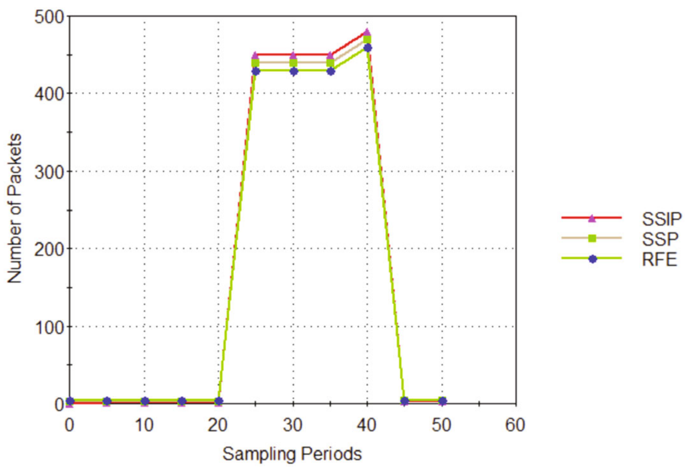

FIGURE 9 Flow trend.[58]

H3 and H4 generate regular TCP, UDP, and ICMP network traffic during the training sample phase. Utilizing the classic DoS attack tool Hping3, anomalous network traffic is generated. Users can receive and transmit data packets by describing the binary or string representation of the packets using the scripting language TCL. The advantage of Hping3 is its ability to customize portions of the packet, allowing users to attack and detect targets based on their specific requirements. In addition, it can perform various tasks with only a few lines of code, including automated security tests with report generation, TCP/IP test suites, NAT-ting, attack simulations, firewall prototypes, and implementation of routing protocols, among others. Figure 8 depicts the case study diagram.

The interval for sampling, denoted as time, is 3 s. The attack occurs during the periods T20 to T45. To collect flow table data from the OpenFlow switch, we sample for 60 periods. Afterwards, we process and normalize the data for each period, resulting in the characteristic values containing both normal samples and DoS attack flow samples. The trends of these characteristic values over the 60 periods are depicted in Figure 9.

In normal circumstances, the size of the data packets is relatively large; however, during an attack, the attacker sends as much continuous data as necessary to accomplish the goal, resulting in relatively small and unchanged data packets.

TABLE 7 Experiment of three kinds of attack.

Type of attacks	Avg. detection rate
TCP	96.83%
UDP	95.24%
ICMP	93.65%

Hence, these two characteristic parameters show significant and noticeable fluctuations during regular periods, but they are relatively small and change gently in the T20–T45 periods. In regular network access, the source host and the destination host produce interactive flow entries. However, during an attack, the use of virtual random source IP addresses and port numbers leads to many requests that the destination host cannot respond to timely, thereby causing a sharp drop in the proportion of interactive flow. Figure 9 illustrates that during the T20 to T45 intervals, the interactive flow entries decrease to almost nothing. Typically, the ratio of interactive flow entries is relatively high and fluctuates within a normal range. The detection rate (DR) and false alarm rate (FAR), determined using the following formulae, show how well an attack is detected.

$$\mathrm{DR} = \frac{\mathrm{DD}}{\mathrm{DD} + \mathrm{DN}} \tag{4}$$

Detection of data flow (DD) signifies that the attack flow has been identified as malicious. In contrast, normal flow detection (DN) indicates that the attack flow has been erroneously categorized as usual.

$$\mathrm{FAR} = \frac{\mathrm{FD}}{\mathrm{FD} + \mathrm{TN}} \tag{5}$$

The flow of detection (FD) denotes that a normal flow has been mistakenly flagged as an attack flow, while TN indicates that a normal flow has been correctly identified as a normal flow.

The experimental study analyzes two categories of network traffic: normal traffic, which includes three standard communication protocols (TCP, UDP, and ICMP), and attack traffic, which involves three different types of malicious attacks utilizing TCP, UDP, and ICMP protocols. The results, presented in Table 7, revealed an impressive overall accuracy rate of 97% for the experiment. Furthermore, it indicates that the combination of SDN and SVM machine learning algorithms yielded superior outcomes, showcasing the effectiveness of this approach.

7 | CONCLUSION & FUTURE DIRECTION

Detection of an anomaly at an early stage in an SDN provides significant leverage in the working efficiency of the network. The recent advances in machine learning and deep learning help in efficient anomaly detection and provide a better quality of service. One such domain of machine learning is GNN. The graph structure offers an appropriate and organized way to represent a network. Here, we have explored the combination of SDN and GNN for anomaly detection and provide a comprehensive overview of network topology from a higher-level perspective. First, we discuss the importance of anomaly detection in modern networks and the limitations of the existing approaches. Here, we provide the taxonomy to bifurcate the various anomaly detection techniques, highlighting the current methods that utilize either SDN, GNN, or hybrid approaches. We represent their underlying principle, application scenarios, benefits, and limitations. In further, we have also provided a comprehensive summary of those approaches. Next, we present the challenges and issues of the existing system and the challenges the integrated SDN-GNN model needs to cater. We also represent the available tools and technologies that can be used for implementing the proposed amalgamation, including popular SDN platforms and GNN libraries. Finally, we also provided a case study to demonstrate the potential of SDN-GNN integration for network anomaly detection, showcasing the benefits of combining SDN's programmability and GNN's ability to capture complex graph-based relationships.

Further research and development are required to realize this integration's full potential. Nevertheless, the proposed amalgamation holds excellent promise and presents an exciting direction for future research in network anomaly detection using integrated approaches. Utilizing the graph structure for network learning will enhance the network's performance and efficiency. Leveraging varied datasets to improve model generalization, including dynamic context-dependent features, and evaluating the suggested technique in actual network contexts will help improvise the traditional models. Further research should focus on examining novel methods for anomaly detection, tackling scalability issues, and merging SDN and GNN with sophisticated security procedures and attacks.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Jitendra Bhatia https://orcid.org/0000-0002-2375-5057
Sudeep Tanwar https://orcid.org/0000-0002-1776-4651

REFERENCES

1. Radford BJ, Apolonio LM, Trias AJ, Simpson JA. Network traffic anomaly detection using recurrent neural networks. arXiv preprint arXiv:1803.10769. 2018.
2. Saeed R, Qureshi S, Farooq MU, Zeeshan M. Sdn/nfv enabled security for an enterprise network using commodity hardware. Paper presented at: 2022 International Conference on Computing, Electronics & Communications Engineering (iCCECE), IEEE. 2022:25–30.
3. Bhatia J, Kakadia P, Bhavsar M, Tanwar S. Sdn-enabled network coding-based secure data dissemination in vanet environment. *IEEE Internet Things J*. 2020;7(7):6078-6087. doi:10.1109/JIOT.2019.2956964
4. Wang S, Balarezo JF, Kandeepan S, Al-Hourani A, Chavez KG, Rubinstein B. Machine learning in network anomaly detection: a survey. *IEEE Access*. 2021;9:152379-152396.
5. Chaudhary A, Mittal H, Arora A. Anomaly detection using graph neural networks. Paper presented at: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), IEEE. 2019:346–350.
6. Omar S, Ngadi A, Jebur HH. Machine learning techniques for anomaly detection: an overview. *Int J Comput Appl*. 2013;79(2):33-41.
7. Lazar V, Buzura S, Iancu B, Dadarlat V. Anomaly detection in software defined wireless sensor networks using recurrent neural networks. Paper presented at: 2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE. 2021:19–24.
8. Lin H, Chengwen W, Masdari M. A comprehensive survey of network traffic anomalies and ddos attacks detection schemes using fuzzy techniques. *Comput Electr Eng*. 2022;104:108466.
9. Xueyuan Duan YF, Wang K. Network traffic anomaly detection method based on multi-scale residual classifier. *Comput Commun*. 2023;198:206-216.
10. Ahmed M, Mahmood AN, Jiankun H. A survey of network anomaly detection techniques. *J Netw Comput Appl*. 2016;60:19-31.
11. Fernandes G, Rodrigues JJPC, Carvalho LF, Al-Muhtadi JF, Proença ML. A comprehensive survey on network anomaly detection. *Telecommun Syst*. 2019;70:447-489.
12. Bhuyan MH, Bhattacharyya DK, Kalita JK. Network anomaly detection: methods, systems and tools. *Ieee Commun Surv Tutor*. 2013;16(1):303-336.
13. Ahmad I, Namal S, Ylianttila M, Gurtov A. Security in software defined networks: a survey. *IEEE Commun Surv Tutor*. 2015;17(4):2317-2346.
14. Chalapathy R, Chawla S. Deep learning for anomaly detection: a survey. arXiv preprint arXiv:1901.03407. 2019.
15. Gupta R, Kumari A, Tanwar S, Kumar N. Blockchain-envisioned softwarized multi-swarming uavs to tackle covid-i9 situations. *IEEE Netw*. 2021a;35(2):160-167. doi:10.1109/MNET.011.2000439
16. Xia W, Wen Y, Foh CH, Niyato D, Xie H. A survey on software-defined networking. *IEEE Commun Surv Tutor*. 2014;17(1):27-51.
17. Fei H, Hao Q, Bao K. A survey on software-defined network and openflow: from concept to implementation. *IEEE Commun Surv Tutor*. 2014;16(4):2181-2206.
18. Zhang Y, Cui L, Wang W, Zhang Y. A survey on software defined networking with multiple controllers. *J Netw Comput Appl*. 2018;103:101-118.
19. Shukla PK, Maheshwary P, Subramanian EK, Shilpa VJ, Varma PRK. Traffic flow monitoring in software-defined network using modified recursive learning. *Phys Commun*. 2023:57:101997.
20. Kumari A, Gupta R, Tanwar S, Kumar N. A taxonomy of blockchain-enabled softwarization for secure uav network. *Comput Commun*. 2020;161:304-323.
21. Gupta R, Tanwar S, Kumar N. Blockchain and 5g integrated softwarized uav network management: architecture, solutions, and challenges. *Phys Commun*. 2021b;47:101355.
22. Gupta R, Patel MM, Tanwar S, Kumar N, Zeadally S. Blockchain-based data dissemination scheme for 5g-enabled softwarized uav networks. *IEEE Trans Green Commun Netw*. 2021c;5(4):1712-1721. doi:10.1109/TGCN.2021.3111529
23. Bruno Astuto A, Nunes MM, Nguyen X-N, Obraczka K, Turletti T. A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutor*. 2014;16(3):1617-1634.

24. Vora J, Kaneriya S, Tanwar S, Tyagi S. Performance evaluation of sdn based virtualization for data center networks. Paper presented at: 2018 3rd International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU). 2018:1–5. doi:10.1109/IoT-SIU.2018.8519846
25. Rusek K, Suárez-Varela J, Almasan P, Barlet-Ros P, Cabellos-Aparicio A. Routenet: leveraging graph neural networks for network modeling and optimization in sdn. *IEEE J Sel Areas Commun*. 2020;38(10):2260-2270.
26. Rafiq A, Khan TA, Afaq M, Song W-C. Service function chaining and traffic steering in sdn using graph neural network. Paper presented at: 2020 International Conference on Information and Communication Technology Convergence (ICTC), IEEE. 2020 500–505.
27. Zhou J, Cui G, Shengding H, et al. Graph neural networks: a review of methods and applications. *AI Open*. 2020;1:57-81.
28. Hodo E, Bellekens X, Hamilton A, Tachtatzis C, Atkinson R. Shallow and deep networks intrusion detection system: a taxonomy and survey. arXiv preprint arXiv:1701.02145. 2017.
29. da Costa KAP, Papa JP, Lisboa CO, Munoz R, de Albuquerque VHC. Internet of things: a survey on machine learning-based intrusion detection approaches. *Comput Netw*. 2019;151:147-157.
30. Ucci D, Aniello L, Baldoni R. Survey of machine learning techniques for malware analysis. *Comput Secur*. 2019;81:123-147.
31. Tahsien SM, Karimipour H, Spachos P. Machine learning based solutions for security of internet of things (iot): a survey. *J Netw Comput Appl*. 2020;161:102630.
32. Hussain F, Hussain R, Hassan SA, Hossain E. Machine learning in iot security: current solutions and future challenges. *IEEE Commun Surv Tutor*. 2020;22(3):1686-1721.
33. Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: research developments, trends and challenges. *J Netw Comput Appl*. 2020;153:102526.
34. Sebbar A, Zkik K, Baddi Y, Boulmalf M, El Kettani MDE-C. Mitm detection and defense mechanism cbna-rf based on machine learning for large-scale sdn context. *J Ambient Intell Humaniz Comput*. 2020;11(12):5875-5894.
35. Khamaiseh S, Serra E, Li Z, Dianxiang X. Detecting saturation attacks in sdn via machine learning. Paper presented at: 2019 4th International Conference on Computing, Communications and Security (ICCCS), IEEE. 2019:1–8.
36. Akbaş MF, Güngör C, Karaarslan E. Usage of machine learning algorithms for flow based anomaly detection system in software defined networks. In: Kahraman C, Onar SC, Oztaysi B, Sari IU, Cebi S, Cagri Tolga A, eds. *Intelligent and Fuzzy Techniques: Smart and Innovative Solutions*. Springer International Publishing; 2021:1156-1163.
37. Bhatia J, Dave J, Bhavsar M, Tanwar S, Kumar N. Sdn-enabled adaptive broadcast timer for data dissemination in vehicular ad hoc networks. *IEEE Trans Veh Technol*. 2021;70(8):8134-8147. doi:10.1109/TVT.2021.3092065
38. Santos R, Souza D, Santo W, Ribeiro A, Moreno E. Machine learning algorithms to detect ddos attacks in sdn. *Concurrency Computat: Pract Exper*. 2020;32(16):e5402.
39. Dehkordi AB, Soltanaghaei MR, Boroujeni FZ. The ddos attacks detection through machine learning and statistical methods in sdn. *J Supercomput*. 2021;77(3):2383-2415.
40. Simpson KA, Rogers S, Pezaros DP. Per-host ddos mitigation by direct-control reinforcement learning. *IEEE Trans Netw Serv Manag*. 2019;17(1):103-117.
41. Sampaio LSR, Faustini PHA, Silva AS, Granville LZ, Schaeffer-Filho A. Using nfv and reinforcement learning for anomalies detection and mitigation in sdn. Paper presented at: 2018 IEEE Symposium on Computers and Communications (ISCC), IEEE. 2018:432–437.
42. Han Y, Rubinstein BIP, Abraham T, et al. Reinforcement learning for autonomous defence in software-defined networking. Paper presented at: International Conference on Decision and Game Theory for Security, Springer. 2018:145–165.
43. Kim T-Y, Cho S-B. Web traffic anomaly detection using c-lstm neural networks. *Expert Syst Appl*. 2018;106:66-76.
44. Liu S, Qiang Q, Wang S. Heterogeneous anomaly detection in social diffusion with discriminative feature discovery. *Inform Sci*. 2018;439:1-18.
45. Aggrawal N, Arora A. Visualization, analysis and structural pattern infusion of dblp co-authorship network using gephi. Paper presented at: 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), IEEE. 2016:494–500.
46. Sharma V, Kumar R, Cheng W-H, Atiquzzaman M, Srinivasan K, Zomaya AY. Nhad: neuro-fuzzy based horizontal anomaly detection in online social networks. *IEEE Trans Knowl Data Eng*. 2018;30(11):2171-2184.
47. Monti F, Boscaini D, Masci J, Rodola E, Svoboda J, Bronstein MM. Geometric deep learning on graphs and manifolds using mixture model cnns. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017:5115–5124.
48. Prasanta Gogoi DK, Bhattacharyya BB, Kalita JK. Mlh-ids: a multi-level hybrid intrusion detection method. *Comput J*. 2014;57(4):602-623.
49. Pajouh HH, Javidan R, Khayami R, Dehghantanha A, Choo K-KR. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks. *IEEE Trans Emerg Top Comput*. 2016;7(2):314-323.
50. Li L, Yang Y, Bai S, Hou Y, Chen X. An effective two-step intrusion detection approach based on binary classification and *k*-nn. *IEEE Access*. 2017;6:12060-12073.
51. Yao H, Wang Q, Wang L, Zhang P, Li M, Liu Y. An intrusion detection framework based on hybrid multi-level data mining. *Int J Parallel Program*. 2019;47(4):740-758.
52. Çavuşoğlu Ü. A new hybrid approach for intrusion detection using machine learning methods. *Appl Intell*. 2019;49(7):2735-2761.
53. Gao X, Shan C, Changzhen H, Niu Z, Liu Z. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*. 2019;7:82512-82521.
54. Tama BA, Comuzzi M, Rhee K-H. Tse-ids: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access*. 2019;7:94497-94507.
55. Golrang A, Golrang AM, Yayilgan SY, Elezaj O. A novel hybrid ids based on modified nsgaii-ann and random forest. *Electronics*. 2020;9(4):577.

56. Liu C, Zhaojun G, Wang J. A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning. *IEEE Access*. 2021;9:75729-75740.
57. Yang Z, Liu X, Li T, et al. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Comput Secur*. 2022;116:102675.
58. Ye J, Cheng X, Zhu J, Feng L, Song L. A ddos attack detection method based on svm in software defined network. *Secur Commun Netw*. 2018;2018:1-8.

How to cite this article: Dhadhania A, Bhatia J, Mehta R, Tanwar S, Sharma R, Verma A. Unleashing the power of SDN and GNN for network anomaly detection: State-of-the-art, challenges, and future directions. *Security and Privacy*. 2024;7(1):e337. doi: 10.1002/spy2.337