Flow-Based Intrusion Detection on Software-Defined Networks: A Multivariate Time Series Anomaly Detection Approach

Sultan ZAVRAK (sultanzavrak@duzce.edu.tr)
Duzce University: Duzce Universitesi https://orcid.org/0000-0001-6950-8927
Murat İskefiyeli
Sakarya University: Sakarya Universitesi

Research Article

Keywords: Intrusion detection, anomaly detection, deep learning, semi-supervised learning, software-defined networks, time series anomaly detection

Posted Date: May 16th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1141416/v3

FLOW-BASED INTRUSION DETECTION ON SOFTWARE-DEFINED NETWORKS: A MULTIVARIATE TIME SERIES ANOMALY DETECTION APPROACH

Sultan ZAVRAK[1] and Murat ISKEFIYELI[2]

Sultan ZAVRAK

[1]Department of Computer Engineering, Duzce University, Duzce/TURKEY

sultanzavrak@duzce.edu.tr

Murat ISKEFIYELI

[2]Department of Computer Engineering, Sakarya University, Sakarya/TURKEY

miskef@sakarya.edu.tr

***Corresponding Author:**

Sultan ZAVRAK

Department of Computer Engineering,
Faculty of Engineering, Duzce University, 81620, Duzce / TURKEY
Phone: +90 (380) 542 1036
E-mail: sultanzavrak@duzce.edu.tr

FLOW-BASED INTRUSION DETECTION ON SOFTWARE-DEFINED NETWORKS: A MULTIVARIATE TIME SERIES ANOMALY DETECTION APPROACH

Sultan Zavrak[1] and Murat Iskefiyeli[2]

[1] Department of Computer Engineering, Duzce University, Duzce, Turkey

[2] Department of Computer Engineering, Sakarya University, Sakarya, Turkey

Abstract

In this study, the SAnDet architecture, which can do anomaly-based intrusion detection by taking advantage of the capabilities offered by the SDN architecture, is presented and implemented as a controller application. A detailed description of this system which consists of three main modules which are statistics collector, anomaly detector, and anomaly prevention is given. More specifically, Replicator Neural Networks (RNN) which is a special variant of the autoencoder, and the EncDecAD method which is a special type of LSTM network that can produce successful results, especially in given data series, are used to identify unknown attacks using flow features collected from OpenFlow switches. In experiments, flow-based features extracted from network traffic data including different types of attacks, are given as input into models as time series. The results of the methods are calculated using the ROC and AUC metrics. Experimental results show that EncDecAD outperforms RNN. Moreover, it is demonstrated that this study has several benefits over previously conducted research.

Keywords: Intrusion detection, anomaly detection, deep learning, semi-supervised learning, software-defined networks, time series anomaly detection

1. Introduction

Cyber-attacks are a set of malicious activities that hinder, obstruct, or destroy information and services that exist on computer networks. Intrusion Detection and Prevention Systems (IDPS) are the mechanisms that perform the functions of detecting and mitigating or preventing these attacks at the network level. The fundamental principle of attack detection is based on the hypothesis that unauthorized behaviors are prominently distinct from normal ones and thus they are detectable. In addition to the many attack detection approaches in the literature, researchers are now focusing more on anomaly-based network intrusion detection methods, since they can identify unknown attacks as well as known attacks [1].

Software-Defined Networking (SDN) is an innovative network paradigm that promises to transform the limitations of currently used traditional network infrastructure [2]. This paradigm takes the control logic of the network from forwarding devices such as routers and switches and places it on the logically centralized controller [3]. From a network security perspective, SDN defines a single control point for forwarding data flows across the entire network infrastructure. By preparing a high-level reactive security monitoring, analysis, and response system, SDN architecture can be utilized to increase network security. [4], [5]. With network security applications implemented with the OpenFlow (OF) protocol, which is an implemented version of the SDN paradigm, it is also possible to implement flows that can handle complex logic rather than simply allow or block [6].

Network traffic analysis or anomaly detection methods frequently generate security-related data transmitted to the central controller. On the controller, applications can be run to analyze and associate the feedback gathered across the entire network. New or updated security policies can be deployed to network components in the form of flow rules based on the results of the analysis. This unified approach can effectively accelerate and protect against security threats to the network [4]. As OF security applications, intrusion detection and intrusion prevention or mitigation methods can be implemented.

The main contributions of the study are summarized as follows:

- This study focuses on the detection of network attacks from flow-based features based on an anomaly-based approach to SDN environments. More specifically, the SAnDet (a shorthand for SDN Anomaly Detector) architecture, which is designed to detect intrusions by taking advantage of the facilities offered by the SDN architecture, is presented and implemented as a controller application. A detailed description of SAnDet consisting of three main modules which are statistics collector, anomaly detector, and anomaly prevention is given.
- The anomaly detectors RNN and EncDecAD are built using a semi-supervised learning approach. In addition, unlike the other studies, this one is unique in that it uses deep learning techniques to detect intrusions by feeding flow-based data as a multivariate time series.
- Based on AUC and accuracy findings, it is proved that EncDecAD-based anomaly detection outperforms RNN. Furthermore, this study has been shown to have several advantages over previous research.

The following is a breakdown of how the article is structured. The previous works on flow-based attack detection in SDN environments are discussed in the next section. The third section contains theoretical information regarding the research methodologies. The fourth section contains the experimental methods and results. The final section explains the results.

2. Related work

The goal of the anomaly detection technique is to create a statistical model that can be used to characterize typical traffic patterns. [7]. In such a situation, any deviation from this pattern is counted as an anomaly and identified as an attack. There have been several studies that have investigated the effectiveness of anomaly-based attack detection in SDN environments. In this section, a summary of the studies in the literature is given in five categories, taking into account the taxonomy done by Jafarian et al. [7], flow-count-based schemes, information theory-based schemes, entropy-based schemes, deep learning-based schemes, and hybrid schemas. A more in-depth look at studies on anomaly-based intrusion detection on SDN networks can be found in [8] and [7].

Traffic flows in the network are initially obtained and aggregated according to a subnet prefix in flow counting-based anomaly detection methods. Although this process brings extra load to the system, it accurately detects abnormal conditions in the network [7]. Zhang et al. [9] suggested a prediction-based scheme that dynamically alters the level of detail of the measurement across both spatial and temporal dimensions to better balance

surveillance overhead and anomaly detection accuracy to detect anomalies. Ha et al. [10] provide a traffic sampling strategy for an intrusion detection system capable of operating on large-scale SDN networks that maximize the usage of malicious traffic's control capacity while keeping the sampled traffic's overall aggregation volume below the control computing capacity. Granby et al. [11] designed SDNPANDA, a plug-in software package to identify anomalies in a software-defined data center. In DoS attacks, Hommes et al. [12] investigate the flow table saturation problem of OF switches. Attack events are determined in the study by assessing the variation in the network's logical topology for various attack kinds and measuring the distance requirements on a table. Using traffic analysis of packet flows, Carvalho et al. [13] demonstrate a real-time anomaly detection and prevention system in SDN. The digital signature of flows acquired using the OF protocol is created by profiling normal traffic behavior. Existing traffic is compared to a previously developed traffic profile to identify any suspicious traffic events that deviate from expected behavior. As a result, reporting is done to verify prevention and results in the event of an attack. He et al. [14] present an anomaly detection scheme in SDN by selecting the required features of the data set and using a density peak-based clustering algorithm. Carvalho et al. [15] present an ecosystem based on SDN to proactively identify anomalies by scanning network traffic. Peng et al. [16] propose a preprocessing module that normalizes flow property vectors to detect DDoS attacks in SDN based on central control, as well as a scheme that processes these vectors and detects anomalies using the KNN method.

Information theory methods are based on the assumption that an existing anomaly in traffic causes a change in the information of traffic data sets. Mehdi et al. [17] focus on traffic anomaly detection in SDN environments. Multiple anomaly detection algorithms in SOHO environments have been experimentally tested to verify their applicability. The study provides experimental results on the efficiency of TRW-CB [18], Rate-Limiting [19], Maximum Entropy Detector [20], and NETAD [21] intrusion detection algorithms using only low network traffic speeds. Dotcenko et al. [6] proposed a method that uses fuzzy logic as well as a mechanism for analyzing network traffic to classify network traffic into attack traffic and normal traffic. Kokila et al. [22] conducted a study examining different machine learning techniques for identifying DDoS attacks in SDN environments and their performance in intrusion detection. In SDN networks, Sathya and Thangarajan [23] investigate gathering data and applying the Decision Tree approach for anomaly detection using the NSL-KDD data set rather than the intended properties of the network traffic.

In determining the randomness of a data set, entropy-based anomaly detection algorithms have proven to be particularly beneficial in spotting network anomalies [7], [24]. Wang et al. [25] proposed a method that could be applied to low-layer switches of SDNs such as OpenvSwitch to detect low-throughput DDoS flooding attacks based on the entropy level. On the other hand, Giotis et al. [26] offer a scalable and effective mechanism for anomaly detection and mitigation in SDN architectures. Unlike the study in [17], this study offers a mechanism that uses sFlow [27] monitoring data instead of OF statistics, which reduces the controller process load and therefore works at higher line speeds, and gives their experimental results. Besides, when an anomaly is detected, it is shown that the network anomaly is successfully reduced (mitigation) by making modifications in the flow tables using the OF protocol. Also, when the sampled statistical data are given as an input to the anomaly detection algorithm, it is seen that the anomaly detection rate is lower than [17]. Francois and Festor [28] capture the flow inputs of each device by the SDN controller, such as switches for anomaly detection in

SDN. After the attack has been identified and the characteristics of the attack packets have been discovered, surveillance will be undertaken on the switch where the attack has been detected, depending on the advantages offered by the controller.

Deep learning methods include a knowledge-based learning-based development algorithm, and the keywords for deep learning are unsupervised machine learning, multi-layer learning, and artificial intelligence [29][7]. Instead, the deep learning structure has a solid capability for adaptability in SDNs in terms of its features and the ability to learn process data on its own [30]. Dey and Rahman [30] propose a network breach detection system based on deep learning for SDN environments. They used the ANOVA F-Test and the REF feature selection scheme to show that the Gated Recurrent Unit LSTM is the best classifier based on several performance assessment metrics. Three components are proposed by Niyaz et al. [31] for a DDoS detection system for SDN environments. A deep learning method is used in the proposed system for feature selection and traffic regulation. A deep learning-based solution for flow-based anomaly identification in SDN is proposed by Tang et al. [32]. To detect suspicious flows in SDN, Garg et al. [33] suggest an anomaly detection method based on a hybrid and real-time constrained Boltzmann machine and gradient decay-based SVM. For SDN-based 5G networks, Li et al. [34] propose an intelligent hybrid IDS. In the suggested system, the k-means algorithm was used for flow classification in addition to the random forest method for feature selection.

In addition to all the aforementioned methods, hybrid schemes have been proposed in the literature for anomaly-based violation detection in SDN networks. For SDN, Santos Da Silvo et al. [35] propose an anomaly detection and traffic classification based on machine learning techniques. Anomaly detection and classification is the lightweight phase based on entropy analysis with low computational cost to instantly detect possibly malicious flows, and heavyweight using SVM (Support Vector Machine) to categorize such flows according to their abnormal behavior.) is carried out in two complementary stages, one stage. Pang et al. [36] propose a new high throughput and highly accurate anomaly detection scheme FADE in SDN. FADE generates a few custom flow rules on these flows to precisely measure its statistics and commands the loading and timeout of these reserved flow rules. Cui et al. [37] propose a scheme that includes four modules in SDN to overcome DDoS: attack detection trigger, attack detection, traceback, and attack prevention. In the study conducted by Braga et al. [38], the forwarding plane of the network device is managed using OF, and the flow statistics are collected, focusing only on Distributed Denial of Service (DDoS) attacks on the data plane, and the performance analysis presented is restricted to the suggested attack detection method and does not include information with entire performance of the system. The method used in the detection of attacks is Self Organizing Maps (SOM), a type of unsupervised artificial neural network. The parameters input to this method are traffic flow statistics.

3. Theoretical Background

This section focuses on the fundamental components of the software-defined networking (SDN) architecture and how it addresses the issues that are present in traditional networks. In addition to that, theoretical details concerning the RNN and EncDecAD anomaly detection methods are also presented.

3.1. Software-Defined Networking

As stated by the Open Networking Foundation (ONF) [39], SDN is a new architecture that separates network forwarding and control tasks. This allows network control to be directly programmable and the architecture for network services and applications to be conceptualized. Infrastructure devices operate as simple forwarding engines in this architecture, dealing with incoming packets according to several rules that are instantaneously produced by a controller in the control layer along with the pre-described program logic. The controller typically executes on a distant machine and communicates with the forwarding elements over a secure link utilizing a few standardized commands. For SDN, ONF offers a high-level architecture [40] that is divided vertically into three main essential layers: i) Infrastructure Layer consists of forwarding elements, which include physical and virtual switches that can be accessed with an open interface. ii) The Control Layer is composed of a collection of software-based SDN controllers that provide unified control capabilities via open APIs for handling forwarding behavior. Controllers can communicate with one another via three communication interfaces: Southbound, Northbound, and East/Westbound. iii) The Application Layer mainly comprises end-user apps that make use of SDN communication and network services. The main parts of this architecture such as the control layer, application layer, infrastructure layer, and communication interfaces between these three layers are shown in detail in Figure 1.

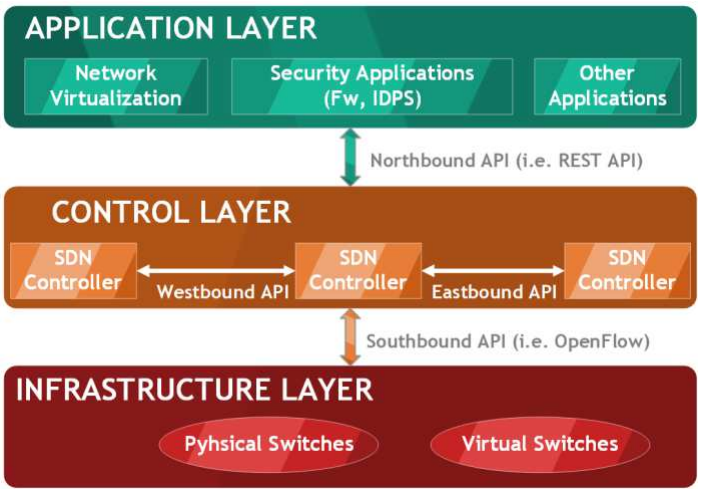

Figure 1. SDN architecture [41]

Through three open interfaces, the SDN controller communicates with these three layers: a) The southbound interface enables the communication between the controller and the forwarding components of the infrastructure layer. The OF protocol, which is managed by ONF, is an essential element for generating SDN solutions according to ONF and can be seen as an encouraging implementation of such an interaction. b) The northbound interface enables the controllers to be programmable by exposing the controllers' universal network abstraction and other features for usage by programs at the application layer. Rather than a protocol, it is viewed as a software API that enables the programming and management of the network. While there is no standardization effort for this, many brands offer REST-based APIs for applications to use to provide a programming interface to their controllers. c) East / Westbound interface, considered a communication interface, is not backed by a recognized standard yet. This is primarily intended to allow inter-controller communication to synchronize the situation for high availability.

Forwarding elements (usually switches) are required to verify a southbound API to be useful in SDN architecture. OF switches come in two types: Software-based (eg Open vSwitch) and hardware-based implementations. Software switches are generally well designed and contain all the features. However, even the latest implementations suffer from being slow. Hardware-based OF switches are usually implemented as ASICs. Although they offer line speed forwarding for a large number of ports, unlike software implementations, they lack flexibility and feature completeness.

The OF-enabled switch can be divided into three main elements [42]. These elements are data path, control path, and OF protocol: a) The data path contains one or more group tables flow tables that search and forward packets. A flow table consists of flow entries associated with actions that tell the switch how to handle the flow. Flow tables are often created by the controller and enable the controller to explain alternative methods of transferring flows. b) A control path is a channel that connects the switch to the controller in programming terms. The OF protocol is used to substitute commands and packets across this channel. c) The OF protocol is responsible for interconnecting switches and controllers. It may include information about messages exchanged, packets sent and received, statistics collecting, and actions to be executed in certain flows.

A flow table entry consisting of several fields in an OF-enabled switch can be organized as follows:

a. Matching Fields are used to identify network packets based on their 15-tuple packet header, ingress port, and optional packet metadata. In Figure 2, packet header fields arranged in accordance with OSI L1-4 layers are shown.
b. The priority of flow entry gives precedence to the matching order of the flow entry.
c. The action set shows the specific actions to be executed on the packets when the title matches.
d. Counters are used to keep track of traffic statistics. (The total quantity of bytes and packets in each flow, as well as the time at which the final packet matches the flow).
e. Timeouts define the maximum amount of time or idle time before the switch overrides the flow.

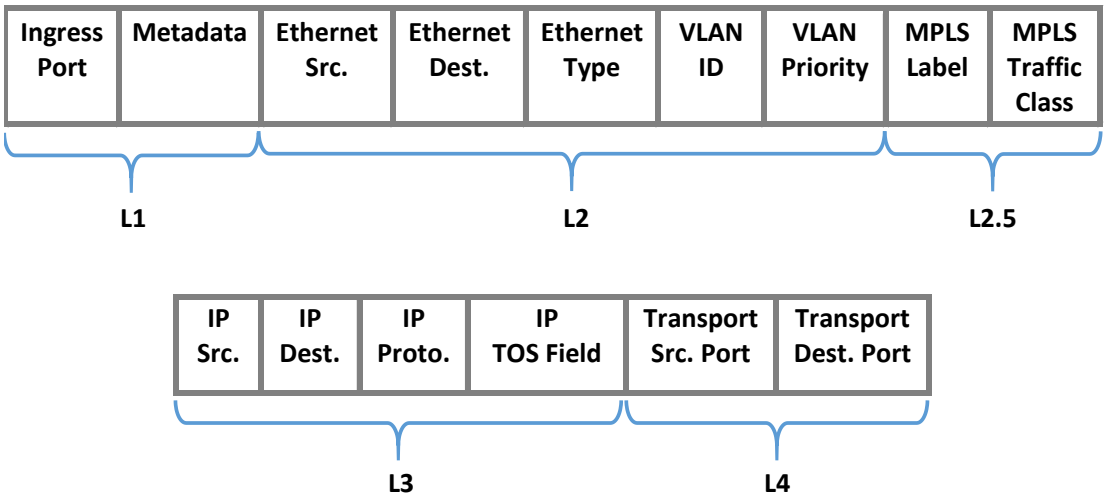

Figure 2. Flow Identification in OF [41]

OF messages can be grouped into three main categories [41]. There are three types of connections: controller-to-switch, asynchronous, and symmetric. Controller-to-switch messages are those initiated by the controller and used to monitor the state of the switches. A switch can initiate asynchronous messages to notify the controller of network events and to modify the switch's state. Finally, symmetric messages are generated automatically by the switch or controller. As soon as an ingress packet arrives at the OF switch, pipeline processing does a scan of the flow tables. The entry into the flow table is determined by matching fields and priority. If the values in the packet correspond to the values in the entry's fields, the packet corresponds to the incoming flow table entry. Any (wildcard field or no field) value in a flow table input field matches all possible values in the header. Only the most critical flow entry should be chosen. If several flow inputs match with the same priority, the chosen flow input is indefinite. To address such a situation, the OF specification offers an optional mechanism that allows for validating whether the new flow input matches the present input. In this way, a packet can be precisely matched to a flow with wildcard fields (macro flow), matched to a flow (micro flow), or not matched to any flow. If the match is located, some actions defined in the match flow table entry are performed. If there is no match, the switch passes the packet (or only the header) to the controller for decision. After checking the related policy in the management plane, the controller responds to the switch to add new entries to the switch's flow table. The switch uses the last input to control both the queued packet and subsequent packets in the same flow.

The controller stands at the heart of SDN networks, connecting applications, and network devices. The SDN controller is responsible for managing all network flows by loading flow entries into switch devices. There are two distinct forms of flow configuration: proactive and reactive. Proactive settings preload flow rules into flow tables. Thus, the flow configuration procedure is completed before the first packet of a flow reaches the OF switch. The primary advantage of a proactive setup is that it reduces the frequency with which the controller is contacted, resulting in a minor installation delay. However, it has the potential to overload switch flow tables. In the reactive setup mode, the controller adds a flow rule to the flow table only when there is no input, which occurs when the first packet of a flow arrives at the OF switch. As a result, communication between the controller and the switch is initiated by a single packet. After a specified period of inactivity, these flow entries are overridden and erased from the table. To respond to the flow setup request, the controller first evaluates the flow to the application's policies and then determines the necessary steps to execute. Following that, it determines a route for this flow and loads new flow entries, including launching requests to each switch along that path.

Transferring information between switches and controllers provides an overview of switch traffic. There are two ways for the switch to provide statistics to the controller. There are two types of flow monitoring: pull-based and push-based. The controller accumulates counters for numerous flows that fit a specified flow specification in the pull-based approach. This technique can optionally generate a report that includes all flows that match a wildcard specification. While this minimizes switch-to-controller traffic, it makes the controller ineffective at learning about the actions of other flows. The pull-based strategy necessitates an improvement in the latency between controller requests, as this can impair the scalability and reliability functions based on statistics collecting. In the push-based approach, statistics are delivered to the controller of each switch to alert it of certain occurrences, such as the creation of a new flow, a timeout, or the deletion of a table entry due to inactivity. Before the input

timeout, this procedure does not notify the controller about the flow's behavior (which indicates that it is unsuitable for scheduling).

3.2. Anomaly Detection

3.2.1. Replicator Neural Network-based anomaly detection

RNNs are neural networks and are specific examples of autoencoders [43] originally proposed as a compression technique [44]. The first study to suggest its use as an anomaly detection technique is recommended by Hawkins et al. [45]. Typically, input vectors in multilayer neural networks are mapped to the target output vectors. However, RNN also uses input vectors for output vectors. In other words, the input values in the output are reproduced by RNN. The RNN's weights are chosen in such a way that the mean squared error is as small as possible. As a result, while standard models are more probable to be successfully replicated by the trained RNN, models characterizing outliers are less accurately represented and have a greater error. Data exclusion is quantified using reconstruction error.

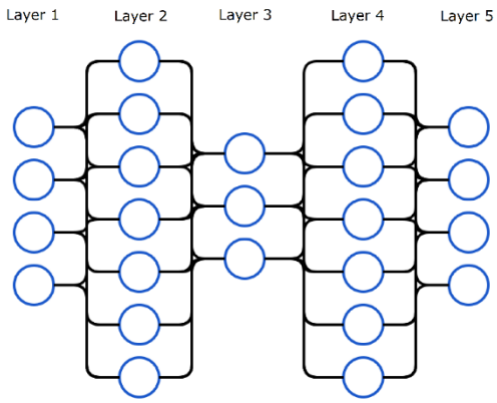

Figure 3. An example diagram of RNN [46]

Cordero et al. propose an approach that uses RNN to identify anomalies in network flows [46]. In this method, an RNN [45] is used primarily to create a model that represents the normal network flow. While it has been demonstrated that the original RNN may be lowered to three layers [47], using the original five layers with the dropout regularization technique [48] produces superior results and avoids overfitting.

Each layer in an RNN is completely connected to every other layer. The activation function of layers 2 and 4 is a nonlinear hyperbolic tangent. The output layer's activation function is linear or identical. The sole distinction between the original RNN and the one used in [46] is in the middle layer's activation function (Layer 3). The original RNN makes use of a stepwise activation function that, in theory, aims to reduce the dimensionality of input data by clustering data samples [45]. While the stepwise activation function possesses intriguing theoretical properties, backpropagation approaches based on gradient reduction do not work adequately with it [46]. Because the gradient components of progressive functions are nearly zero, the learning process stalls. Instead of

this activation function, it employs the sigmoid activation function, which has been shown to be effective as an intermediate activation function for RNNs [49].

The features extracted from different network flows are used to build RNN models. Depending on the tools used, many different features can be extracted from the network for training. The number of selected features is proportional to the number of input neurons. At each training stage, the RNN is fed the extracted flow features as an input. A validation set is used to ascertain the degree to which the learning process is capable of generalization. After training, the RNN can be used as a normal model for the purpose of calculating anomaly scores (ASs). ASs that exceed a predetermined threshold are considered abnormal.

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ represent the set of all N training samples where $\mathbf{x} = \{x^1, x^2, \dots, x^D\}$ is a flow with D features and the function $f(\mathbf{x}) = \hat{\mathbf{x}} + \vec{\epsilon}$ correspond to the output of an RNN. The reconstruction of $\mathbf{x}$ is represented with the vector $\hat{\mathbf{x}}$, and the vector $\vec{\epsilon} = \{\epsilon^1, \epsilon^2, \dots, \epsilon^D\}$ is the error elements of the reconstruction. The weights of the neural network are updated by employing backpropagation with particular gradient descent techniques such as Stochastic Gradient Descent (SGD). In the learning process, the loss function being minimized is formulated in Eq 1.

$$\mathbf{L} = \frac{1}{N}\sum_{i=1}^{N} \left(\mathbf{x}_i - (\hat{\mathbf{x}}_i + \vec{\epsilon}_i)\right)^2 \qquad (1)$$

The network aims to achieve a combination of weights such that $\hat{\mathbf{x}} \approx \mathbf{x}$ and $\epsilon \approx 0$ since the purpose of backpropagation with gradient descent is to minimize $\boldsymbol{L}$. The noise is added throughout the network by randomly detaching units in each learning iteration with the dropout method [48] to evade learning the trivial identity solution $f(\mathbf{x}) = \mathbf{x}$. The residual value $\vec{\epsilon} = \mathbf{x} - \hat{\mathbf{x}}$ is employed to calculate the AS, which determines how anomalous a set of features is. The AS of the set of flow features $\mathbf{x}$ is defined in Eq. 2.

$$AS(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D} (x^i - \hat{x}^i)^2 = \frac{1}{D}\sum_{i=1}^{D} (\epsilon^i)^2 \qquad (2)$$

To determine whether the network flow $\mathbf{x}$ is an anomaly or not, a threshold is selected to decide if the AS is too high for a flow to be counted as normal. The threshold is assigned to the highest reconstruction error E found during training after the elimination of the outliers.

3.2.2. LSTM based encoder-decoder for anomaly detection

LSTM networks [50] are recurrent models used for a variety of learning tasks such as handwriting recognition, speech recognition, and emotion analysis. To map an input sequence to a vector representation of constant dimensionality, an LSTM-based encoder is used. The decoder is another LSTM network that generates the desired sequence using this vector representation.

Malhotra et al. [51] suggest an LSTM-based Encoder-Decoder (EncDecAD) scheme for time series anomaly detection. In this architecture, the encoder generates a vector representation of the input time series, which the decoder uses to reproduce it. The EncDecAD is trained to recreate samples of "normal" time series using the input time series as the output. Following that, the reconstruction error is used to determine the probability of an anomaly occurring at that location. It is demonstrated that using an encoder-decoder model trained on solely normal sequences, anomalies in multivariate time series may be detected. According to this theory, the encoder-decoder has only seen and understood normal examples of the training data during the training phase. Given an abnormal sequence, the trained model fails to reproduce it well by resulting in higher reconstruction errors in contrast with normal sequence reconstruction errors.

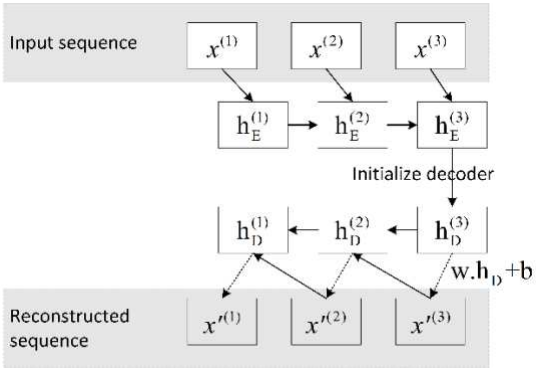

Figure 4. LSTM Encoder-Decoder inference steps

The definition of the EncDecAD approach is mathematically expressed as follows. Given a time series $X = \{x^{(1)}, x^{(2)}, \dots, x^{(L)}\}$ of length L, where each point $\mathbf{x}^{(i)} \in R^m$ is an m-dimensional vector of readings for m variables at a time t_i. The case is explored where such time series are available or can be acquired by selecting a window of length L over a longer time series. To recreate the normal time series, the EncDecAD model is trained. Following that, the reconstruction errors are used to determine the likelihood of a point in a test time series being anomalous to produce an anomaly score $a^{(i)}$ for each point $\mathbf{x}^{(i)}$. A greater score for anomaly indicates that the point is more likely to be abnormal.

To reconstruct examples of normal time series, an LSTM encoder-decoder is trained. The LSTM encoder approximates the input time series with a fixed-length vector representation. Using the current hidden state and the value calculated by the LSTM decoder at the previous time step, this representation is used to reconstruct the time series.

The input $\mathbf{x}^{(i)}$ is used to achieve the state $\mathbf{h}_D^{(i-1)}$ and thenceforth $\mathbf{x}'^{(i-1)}$ corresponding to target $\mathbf{x}^{(i-1)}$ is predicted by the decoder in the training phase. The decoder uses the predicted value $\mathbf{x}'^{(i)}$ as input, and then predicts $\mathbf{x}'^{(i-1)}$ in the inference phase. Given a set of normal training sequences as s_N, the objective of the model training is to minimize the function $\sum_{X \in s_N} \sum_{i=1}^{L} \|x^{(i)} - x'^{(i)}\|^2$. To achieve the encoder's hidden state $\mathbf{h}_E^{(i)}$ at the time t_i, the value $\mathbf{x}^{(i)}$ at the time t_i and the encoder's hidden state $\mathbf{h}_E^{(i-1)}$ at the time $t_i - 1$ are used. The prediction $\mathbf{x}'^{(i)}$ and $\mathbf{h}_D^{(i)}$ is used to achieve the next hidden state $\mathbf{h}_D^{(i-1)}$ by the decoder.

The normal time series is subdivided into four groups as s_N, v_{N1}, v_{N2} and t_N. Additionally, the anomalous time series are divided into two groups as v_A andt_A. The LSTM encoder-decoder reconstruction model is developed using the set of sequences s_N. When the encoder-decoder model is being trained, the set v_{N1} is used for early stopping. The formula of $\mathbf{e}^{(i)} = | \mathbf{x}^{(i)} - \mathbf{x}'^{(i)} |$ is used to calculate the reconstruction error vector for t_i. The parameters $\boldsymbol{\mu}$ and Σ of a normal distribution $\mathcal{N}(\mu, \Sigma)$ are estimated by using the error vectors for the points in the sequences in the set v_{N1} with maximum likelihood estimation. After that, the anomaly score is calculated by $a^{(i)} = \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1} \left(\mathbf{e}^{(i)} - \boldsymbol{\mu}\right)$ for any point $\mathbf{x}^{(i)}$. In a supervised approach, if $a^{(i)}$ is greater than threshold τ, a point in a sequence can be expected to be "anomalous", otherwise "normal". If sufficient anomalous sequences are present, a threshold τ over the probability values is learned to optimize $F_\beta = (1 + \beta^2) \times P \times R / (\beta^2 P + R)$, where R is recall and P is precision. Here, "normal" refers to the negative class, while "anomalous" refers to the positive class. If a window contains an anomalous pattern, the entire window is marked as "anomalous." This method is particularly advantageous in a variety of practical applications where the precise location of the anomaly is unknown. On the validation sequences in v_{N2}, and v_A, the parameters τ and c are determined with maximum F_β.

3.3. Evaluation Metrics

The proposed approach should be validated using a relevant metric. The binary classification results can be classified into four categories [52]: 1) True Positive (TP): Positive instances have been classified accurately; 2) False Negative (FN): Positive instances have been classified incorrectly; 3) False Positive (FP): Negative instances have been classified incorrectly; 4) True Negative (TN): Negative instances have been classified incorrectly. Furthermore, additional metrics can be determined by starting with the prior ones [53]:

Accuracy: This metric is expressed as the proportion of correct predictions to total instances:

$$\text{Accuracy} = (\text{TP}+\text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (3)$$

True Positive Rate (TPR): This metric is equivalent to the proportion of all "correctly identified instances" to all "examples that should be identified".

$$TPR = TP/(TP + FN) \quad (4)$$

False Positive Rate (FPR): This metric denotes the proportion of the "number of misclassified negative instances" to the "total number of negative instances".

$$FPR = FP/(FP + TN) \quad (5)$$

Receiver Operating Characteristics (ROC): In the case of a class imbalance problem in the dataset, the ROC curve [54][55] is being used as a normal criterion for testing classifiers [56]. When faced with an issue of class imbalance, the area under the receiver operating characteristic curve (AUC) metric is frequently utilized as a de facto criterion for evaluating the effectiveness of classifiers. After sorting by classification probabilities, the

AUC can be used to determine how frequently a random instance of a positive class ranks higher than an instance of a negative class.

4. Anomaly-based intrusion detection system for SDN

Some principles should be taken into consideration in SAnDet (SDN Anomaly Detector), which is designed to work in SDN environments and can actively use the opportunities offered by this architecture. In the design of SAnDet, the key principles given in the following were considered taking into account the potentials offered by OF in addition to a few principles determined by Giotis et al. [26]:

a. Separating data collection, anomaly detection, and prevention (mitigation) with a modular design.
b. Compatibility with OF-enabled Layer 2 and Layer 3 devices.
c. Fast anomaly detection and prevention (mitigation) in real-time environments using separate data and control planes.
d. Utilizing OF's capabilities for gathering statistics and mitigating attacks.

The proposed approach is based on a set of 12-tuple flow definitions associated with four specific variables, usually included as a flow entry in the OF switch. These variables are; i) action rule specifying how to forward when any packet matches its associated flow entry, ii) a soft timeout variable in place of the flow being invalidated after the final packet match, iii) the quantity of packets matching that flow since the flow input processing, iv) in the event of a packet match conflict, a specific priority is delegated to each flow input, indicating which flow rule will be determined in the event of an occurrence.

SAnDet's architecture consists of three main modules as illustrated in Figure 5 [26]:

- Statistics Collector Module: The collector module is in charge of collecting the data necessary for flow-based anomaly detection. This module collects flow data on a periodic basis and passes it to the Anomaly Detection module. Two distinct data collection strategies have been described in the literature. The first of these is the OF approach, which works by periodically querying the switch and accumulating the incoming responses. The second is a flow monitoring mechanism that makes use of packet sampling. Giotis et al. use sFlow, which is vendor-independent [26]:
- Anomaly Detection Module: At specified periodic time intervals, the collection module sends data to the Anomaly Detection module. It has been designated as a ten-second time slot. Giotis et al. [26] utilized an algorithm based on entropy. They also mentioned that this module can be utilized with any statistical anomaly detection, machine learning-based anomaly detection, or data mining-based anomaly detection technique.
- Mitigation Module: The anomaly prevention module attempts to mitigate identified attacks or breaches by adding (or modifying) flow entries to the OF switch's flow table in order to prohibit the targeted malicious traffic [26]. These flow entries have a higher priority than other flows in the flow table. Adding certain host-related rules to a whitelist table allows for the avoidance of blocking legal network flows that exhibit anomalous behavior that resembles malicious behavior. Additionally, malicious rules are added to a blacklist (malicious address) table.

Figure 5. SAnDet architecture

4.1. Data collection and feature extraction

The OF approach uses the OF protocol to collect flow statistics from switches. As required by the OF protocol, the controller handles periodic flow statistics requests by aggregating them with the relevant counters for all flow entries in the OF switch [26]. The switches' flow counters are updated only when a forwarding query procedure matches an entry in the flow table. As a result, the gathering of flow statistics in a native OF environment is inextricably linked to the controller's packet forwarding mechanism [26]. In our scenario, when the forwarding logic is dictated by the anomaly detection strategy, layer 3 and 4 protocol fields are utilized. As a result, the flow table's necessary entries are reduced to a single flow, and a single flow entry is formed by evaluating forward and backward directions based on the source IP, source port, destination IP, destination port, and protocol fields. The remainder of the fields in the flows contain wildcards, which allow them to match any value in the fields.

```
active_flows # the active flows in specific time interval
t # time interval to query flow statistics
F # flow statistics table

if e is initialization event then
  active_flows <- 0, U <- 0
end if

if e is PacketIn event then
   f.stats_values <- 0
   f <- new_flow (src_ip,dst_ip,src_port,dst_port,ip_proto,stats_values)
   add f to F
else if e is FlowRemoved event then
  f.stats_values <- e.stats_values
  f.isFlowRemoved <- true
else if e is timeout t event then
   for all flows f1 in F
       # if any flow entry in the statistics table is active on the switch

       if f1.isFlowRemoved is false then
            send a FlowStatsRequest to edge_switch
       end if
   end for
else if e is a FlowStatsReply event for flow f then
   F.f.stats_values = e.stats_values
end if
```

Figure 6. The flow statistics collection algorithm

The OF approach sends a message (FlowRemoved) to the controller when a flow entry is deleted from the switch, along with this information, as well as other data such as counter values of the flow entry. Gathering statistics with the OF approach can be performed when a switch responds (FlowStatsReply) to a flow statistics request (FlowStatsRequest message querying the switch for flow statistics) from the associated OF controller. By using these two messages together, data collection of flow entries can be accomplished [57]. Every new packet that arrives initially is added to a flow table on the controller. Later, messages are expected from these flow entries that are programmed to send a message when the flow entry is deleted at a certain time. If these messages are received from all entries in the table during this period, it is assumed that all flow statistics have been obtained and these data are transferred to the detection algorithm. If there is no deleted message from at least one flow entry, the statistics collection message is sent to the OF switch and a response message is expected. The algorithm of this approach can be given in Figure 6. As a result, the switch reacts to huge portions of the flow table's content. Each stack includes a subset of the flow entries, as well as packet counters for each flow. The anomaly detection algorithm, on the other hand, only includes counter contents from the counters as a consequence of the query, such as the number of packets matching each flow entry within a certain timeframe, whereas flow table records' packet counters contain the total number of packets after each rule was instantiated. Therefore, to find the number of packets corresponding to the specified time window, a record of the status of the flow chart for previous time windows must be kept and compared with valid data for each flow entry. Due to the absence of sampling in the OF data collecting process, it is able to collect and analyze in great detail all network traffic streaming through the switch [26]. As a result, this approach has proven to be effective in monitoring networks of low to medium traffic volumes [38][17]. The derived features in Table 1 are also calculated for the flow, which is the corresponding pair of a flow, and added separately to the data set. In Table 1, *P, B, n,* and *D* corresponds to the number of packets, the number of bytes, the number of flows, and active duration, respectively.

Table 1. Flow features

	Feature Name	**Description**	**Flow direction**	
			Forward	**Backward**
Flow Id	Source IP			
	Destination IP			
	Protocol type	TCP, UDP, and ICMP		
	Type of Service			
	Source Port			
	Destination port			
Basic features	Byte count		✓	✓
	Packet count		✓	✓
	Active duration in seconds		✓	✓
	Packet rate in seconds		✓	✓
	Byte rate in seconds		✓	✓

Derived Features1 [38], [58]	The average number of packets per flow (APf)	$APf = \frac{1}{n}\sum_{i=1}^{n} P_i$	✓	✓
	The average number of bytes per flow (ABf)	$ABf = \frac{1}{n}\sum_{i=1}^{n} B_i$	✓	✓
	Average active duration per flow (ADf)	$ADf = \frac{1}{n}\sum_{i=1}^{n} D_i$	✓	✓
	The growth of single flows (GSf)	$GSf = \frac{Num\_Flows - (2 * Num\_Pair\_Flows)}{interval}$		
	The growth of different ports (GDP)	$GDP = \frac{Num\_Ports}{interval}$		
	The number of flows from the same source IP			

4.2. Attack detection and prevention

A method should be developed that takes the features extracted from the flows as input and detects whether there is an attack on the network. The deep learning-based methods, which are popular recently and have contributed significantly to the solution of many problems, have been preferred as the detection method, and their performance has been examined.

```
black_list # blacklist flows
detected_flows # identified flows

for all flows f in detected_flows  do
   if black_list.contains(f) then
      do nothing # already in blacklist
   else
      # write drop rule to the switch for f
      create match m object
      m.srcIp = f.srcIp
      m.destIp = f.destIp
      m.srcPort = f.srcPort
      m.destPort = f.destPort
      m.proto = f.procto
      m.action = drop
      m.idleTimeout = default_timeout * delay_ratio
      m.hard_timeout = default_hard_timeout

      write m to the switch
      # add malicious flow to blacklist
      black_list.insert(f)
   end if
end for
```

Figure 7. Anomaly mitigation algorithm

In SAnDet architecture, the focus is to employ anomaly-based techniques which are capable of detecting both known and zero-day attacks. In deep learning methods, RNN and EncDecAD were able to successfully detect anomalies in data sets according to studies in the literature. In addition to the flow statistics collected from the network, the features derived from them are given as input to these two methods and the detection performances are calculated according to certain evaluation criteria. Although AUC is recommended to be used as a criterion in the evaluation of anomaly-based methods, it has been stated that accuracy criteria should also be taken into account in data sets with imbalanced class distribution [56].

5. Performance Evaluation

In this section, performance evaluation is made as to the attack detection and prevention mechanism of OF protocol. Also, the benefits of OF are investigated to prevent identified malicious traffic, by using SDN controller facilities.

Floodlight [59] is an open-source OF controller with a modular architecture. Through the API provided by the Floodlight controller, periodic and aperiodic data collection are performed. After that, the derivation of new features from the basic features of the flow inputs collected, and the implementation of three different components responsible for flow entry modification tasks were performed. The Statistics Collector module, which collects statistics and generates new features, has been implemented. An anomaly detector module has been implemented, where models developed from both RNN and EncDecAD methods can be applied and together with which different detection algorithms can be easily integrated. The results of this action are then sent to the Anomaly Prevention module to provide intrusion countermeasures. This enables the user to create or employ their preferred anomaly detection technique as long as it can forward the required information to the Anomaly Prevention module. Performance assessment trials were conducted with some of the network traffic in the ISCX2012 data set.

5.1. Testbed environment and traffic generation

SAnDet system was implemented as Floodlight controller modules for anomaly detection and prevention functions. OF-enabled switches are required to run experiments. Open vSwitch [60], a software switch capable of handling the necessary traffic loads, is used for this purpose. The experimental environment of the study was provided by Mininet [61] emulation software, which is an open-source tool that is frequently used in the literature. This emulation environment supports OpenFlow vSwitch [60], which is superior to hardware switches in its features and is designed to develop SDN-supported network prototypes. The emulation environment was hosted on a virtual server with 16 GB of RAM, and a quad-core 3 GHz processor. Besides, Floodlight [59] is used as controller software because of the open-source functionality that it provides. Figure 8 shows the experimental setup together with Floodlight modules used to evaluate the two approaches mentioned above. The controller software is run in a virtual machine on the server. Network traffic captured for performance evaluation is injected into a single switch (in this case software-based Open vSwitch). It should be made clear that the proposed mechanism can be applied to multiple OF-enabled switches and more general network topologies with the corresponding prevention rules.

Some parameters need to be set in OF-based statistics collection. In particular, a specific value has to be set for the idle timeout for each flow entry. This is because the collector module requests any flow entry to expire to gather the relevant flow statistics. Therefore, the time window is specified by assigning the statistical collection period to 10 seconds, and the idle timeout to 3 seconds.

Table 2. The number of all flows collected from traffic generated

Day	**Normal**	**Attack type**	
		DoS	**Portscan**
11 June Friday	3452552	0	0
12 June Saturday	557885	210569	252874
16 June Wednesday	2213469	0	0
Total	5820763	463443	

Normal or benign traffic of the ISCX2012 data set is used to evaluate the performance of the attack detection and prevention mechanism. To be more precise, traffic captured on "Friday 11 June" and "Wednesday 16 June" is used as normal network traffic. Besides, benign traffic, which includes the "Saturday" traffic track, is played normally, and specified DoS and port scanning attacks are performed at certain time intervals.

The ISCX2012 dataset [62] was produced in 2012, catching traffic in the network emulation environment for more than a week. The authors describe attack scenarios while α profiles, a dynamic approach to creating an attack detection dataset with normal and malicious network behavior, whereas β profiles describe typical user behavior such as email writing or web browsing. These profiles are used to create a different data set that is packet-based and flow-based in both directions. The dynamic strategy enables the ongoing generation of fresh data sets. Although this data set includes a wide variety of attacks such as SSH brute force, DoS, or DDoS, the traffic traces containing these attacks were not used in this study because it disrupts the attack characteristics of the tcpreplay tool.

Table 3. Number of samples produced from flows collected over time

Day	**Normal**	**Attack type**	
		DoS	**Portscan**
11 June Friday	8640	0	0
12 June Saturday	8275	55	35
16 June Wednesday	8644	0	0
Total	25559	90	

These traffic trace files were used in experiments to assess the accuracy and detection abilities of the OF approach. Replaying captured packet trace data and injecting produced traffic onto a particular Ethernet port is accomplished using the tcpreplay [63] program, which is capable of replaying captured traffic at the rate at which it is captured. To perform the attack, hping3 [64], a programmable software tool, was used to allow sending packet strings with random protocol field values. This enables packets to be sent to a predefined destination IP address and port to perform a DoS attack. Finally, through the hping3 tool, the attack is performed by randomly selecting the source and destination ports with a specific source and destination IP address for the portscan scenario.

Using the flow statistics gathered from the OF switches during the flow collection time interval, the derived features corresponding to that time interval are calculated and added to the data set as a single instance In other words, with the aid of derived features, numerous flow entries during the time interval in which the statistics are collected are aggregated into a single instance in the data set. A similar process was repeated for flows at successive time intervals in order to generate a time series dataset in which each instance corresponds to a time step. Since each instance corresponding to sequential time steps in the dataset comprises several features (derived features), a multivariate time series dataset is produced. The maximum active duration, maximum packet counts, and maximum byte counts for all flows during the specified time interval are also included as a derived feature. The number of all flows collected from the generated traffic is shown in Table 2. Besides, the number of samples containing features derived from the flows collected in a certain time interval (10 seconds) is shown in Table 3.

5.2. Anomaly detection and mitigation

The SSL (Semi-Supervised Learning) strategy has been chosen as a learning strategy because it requires less information, time, and effort, and unlabeled data is easier to obtain for an intrusion detection system than labeled data [65]. To explain the SSL strategy with greater clarity, during the training phase it is known that the dataset comprises only normal examples, which is supervised information (labeled as normal) for the model. In the testing phase, we provide no information about the dataset, and we expect the model to classify the samples as normal or abnormal (the label which is not been trained before) Furthermore, this strategy is better suited to the detection methods' unsupervised training nature.

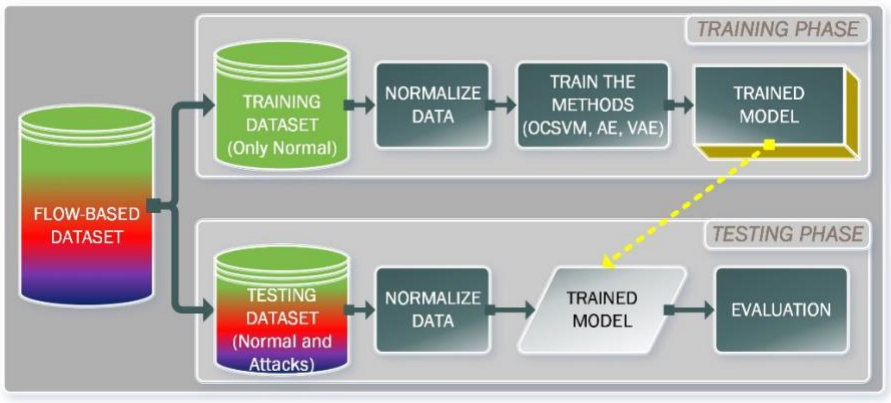

Figure 8. The Semi-Supervised Learning Strategy [65]

As can be seen in Figure 8, the data set is divided into two separate parts as training and test data sets. To create a normal profile of the network traffic by applying the SSL strategy, only the labeled data set containing normal flow characteristics is used in the training phase. The testing process makes use of an unlabeled data set containing both normal and attack flow characteristics.

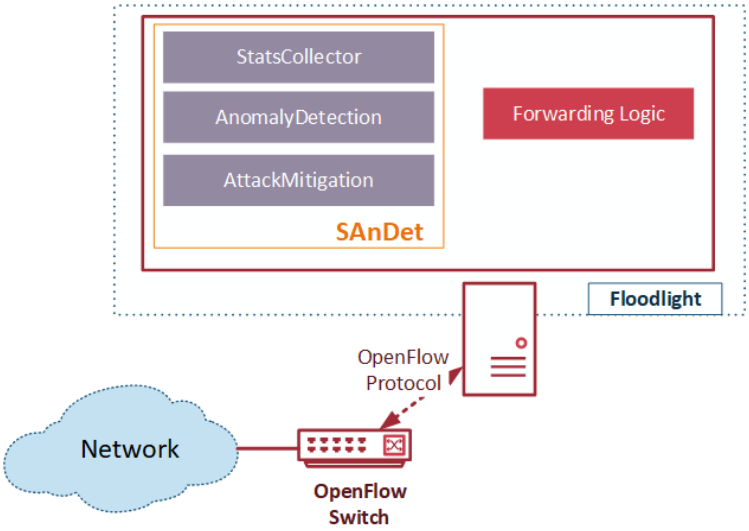

Figure 9. The schematic representation of the experimental environment

Since the techniques used in this analysis are parameterized, the models' performance should be calculated using the right parameters. Because abnormal data is not included in the training process, the cross-validation operation cannot be accomplished in hyperparameter optimization [66]. Thus, the adjustment of the hyperparameters is done mostly taking into account the suggestions mentioned by Patterson and Gibson [67]. Following that, the RNN and EncDecAD hyperparameters are often set by trial and error.

Table 4. The evaluation results of RNN and EncDecAD

Method	**Window Size**	**Bottleneck layer dimension**	**Accuracy**	**AUC**
EncDecAD	6	32	**0.993**	**0.933**

EncDecAD	4	48	**0.993**	0.932
EncDecAD	10	8	0.992	0.931
EncDecAD	7	32	0.992	0.924
EncDecAD	8	32	**0.993**	0.920
EncDecAD	5	96	0.99	0.911
EncDecAD	9	8	0.992	0.911
EncDecAD	3	48	0.992	0.886
RNN	5	32	0.98	**0.87**
RNN	10	64	0.99	0.864
RNN	6	32	0.987	0.820
RNN	3	8	**0.992**	0.818
RNN	8	64	0.99	0.808
RNN	9	48	0.991	0.805
RNN	4	32	0.978	0.792
RNN	7	64	0.991	0.766

The experiments were carried out using a basic deep autoencoder architecture. The architectural diagram of the RNN and the architectural diagram of EncDecAD are shown in Figure 3 and Figure 4, respectively (when the window size is 3). Models were trained and developed using different dimensions such as 8, 16, 32, 48, 64, 80, and 96 in the bottleneck layer of RNN and hidden layers of EncDecAD. The dimensions of the layers in the best performing RNN and EncDecAD models were established by trial and error while maintaining the layer count constant, that is, conforming to the given architectures. The following parameters are utilized in the RNN and EncDecAD neural network configurations: In training trials, learning rates (learning rates) such as [0.1, 0.01, 0.001] are employed. The optimal value for the learning rate is 0.001. Both neural networks were trained using the backpropagation technique and the conjugate gradient optimization approach, which is suggested for big data sets and outputs with real values [67]. The Adam updater [68] was chosen to avoid the local minimum and seek more efficient optimization alternatives. The following additional parameters are utilized in the EncDecAD configuration. In hidden layers, the hyperbolic tangent function is utilized as the activation function. n the hyperparameters used to configure the RNN, which is commonly used and recommended in [46], the sigmoid activation function was used in addition to the output layer, and the mean square error loss function, and the soft-max activation function were used in the output layer. Both RNN and EncDecAD models were implemented with the library [69] based on Pytorch [70] and trained for 50 epochs with 16 batch sizes. Reconstruction error was used as an anomaly score in both RNN and EncDecAD methods.

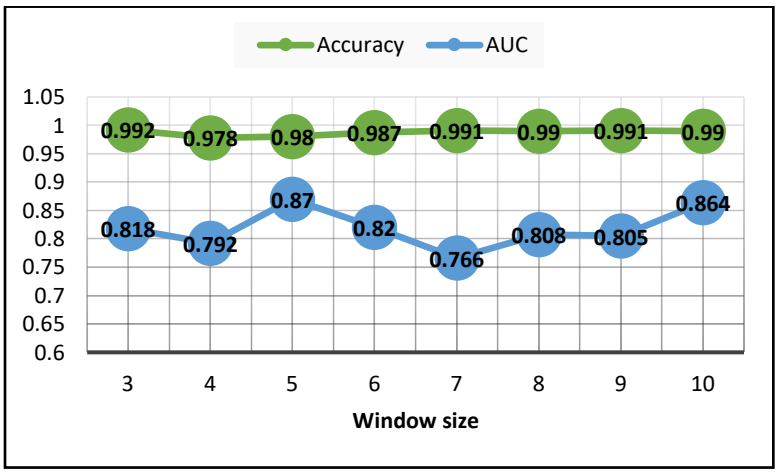

Figure 10. The plot of accuracy and AUC over a time window in RNN

Initially, the normalization process was performed on the traffic traces by using the feature scaling method [71] to bring all values to the range [0,1]. To assess the algorithms' performance, distinct training and test data sets were used. The models were constructed by training the neural network exclusively on "Friday" and "Wednesday" normal flow data. This is because models are developed in an unsupervised manner, and the label for the final column in the dataset corresponds to the attack class that was not used during training. The testing procedure included both normal and attack traffic on alternate days. All attacks classified as "anomalies" during the testing procedure are those that are not "normal" or "benign." Metrics were computed for the performance evaluation of an attack class using only normal flow records and attack class-specific flow data, ignoring any other attack classes.

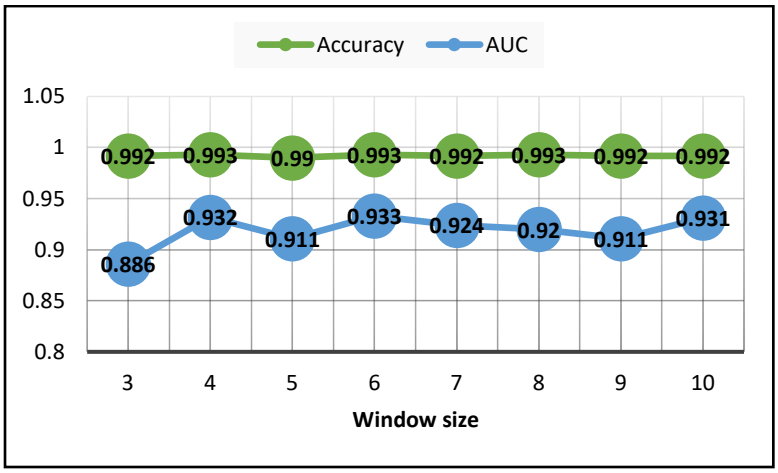

Figure 11. The plot of accuracy and AUC over a time window in EncDecAD

Performance evaluation of the methods is carried out using AUC and accuracy metrics. An assessment based on the AUC criterion is performed as it is the de facto standard for imbalanced anomaly detection and easy interpretation [72]. The AUC results of the methods are given in Table 4. When the results are examined, it is seen that the EncDecAD method gives better results than RNN in different time windows and bottleneck layers as can be seen in Figure 10 and Figure 11. It means that EncDecAD learns the relationship from the sequences better than RNN. The results show that EncDecAD also gives better results compared to studies listed in Table 5 in terms of accuracy and AUC.

Table 5. The comparison of SAnDet with previous studies

Reference	**Method**	**Number of features**	**Attack types**	**Controller**	**Dataset**	**Result**
Niyaz et al. [31]	Stacked Autoencoder	TCP, UDP and ICMP flow features	DDoS	POX	Traffic datasets collected from different environments	Accuracy % 95.65

Tang et al. [32]	Deep Neural Network	6 features	DoS, U2R, R2L, Probes	OF controller	KDD	Accuracy % 75.75
Tang et al. [73]	Recurrent Neural Network	6 features	DoS	POX	NSL-KDD	Accuracy %89
Braga et al. [38]	SOM	6 features	DDoS	NOX	KDD99	Accuracy % 98.57
Kokila et al. [22]	SVM	Flow statistics	DDoS	SDN controller	DARPA	% 95.11
Abubakar et al. [74]	Neural network	7 features	DoS, U2R, R2L, and scan	OpenDaylight	NSLKDD	Accuracy %97
Wang et al. [75]	SVM	Feature reduction	DDoS	Ryu	KDD99	Accuracy % 99
The proposed approach (SAnDet)	EncDecAD (LSTM)	21 features	DoS and Portscan	Floodlight	ISCX2012 as normal traffic and attack with hping3	Accuracy % 99.3 AUC % 93.3

Table 5 shows a performance comparison with previous works. We compare them in terms of methods, number of features, attack types, controller software, dataset, and performance metrics. Similar to previous studies, the proposed detection approach is also trained with normal network traffic and tested with both normal and attack traffic during the test phase. In terms of real-time anomaly detection from a multivariate time series flow-based data, our research differs from previous work. In other words, considering the flow features in a specific time step as multivariate and attempting to detect whether an attack is occurring within a specific time window (which contains multiple time steps) is a contribution that distinguishes this study from others. Another superior aspect of this study is the evaluation of the proposed approach using a more recent data set than previous studies.

On the other hand, while the proposed method detects flows with characteristics that differ from the normal profile as anomalies, the use of fewer attack types in the evaluation phase can be seen as a weakness of the study in contrast to previous studies.

In addition, it is essential to note that comparing the accuracy and AUC results in Table 5 with those of previous studies would not be a fair evaluation, as this study employs a different methodological approach. However, it would be appropriate to announce the results as both accuracy and AUC criteria so that future studies can make comparisons.

6. Conclusion

In this study, an intrusion detection and prevention architecture called SAnDet is proposed and implemented as a controller application. To put it more clearly, a statistics collection module using the OF protocol, an anomaly-based attack detection module that uses the EncDecAD method to detect attacks, and an intrusion prevention module that can prevent attacks via the OF protocol when an attack is detected is implemented as an SDN controller application. By employing a semi-supervised learning strategy, the attack detection capabilities of the RNN and EncDecAD algorithms for anomaly detection were researched. The models were created using solely normal flow-based data. Besides, the tests of the models were carried out using a time series data set containing both normal and abnormal. Experimental results were obtained using ROC curves, AUC, and accuracy metrics. According to the results, it has been observed that EncDecAD has a higher rate of detecting attacks compared to other methods recommended in the literature.

As a future study, it is important to investigate how the method will perform by increasing the types of attacks in the data set. In addition, how the change in the time interval for statistics collection will affect system performance in terms of both controller load and attack detection rate is another future work.

Declaration of Interest Statement

The authors declare that they have no conflict of interest.

REFERENCES

[1] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *Commun. Surv. Tutorials, IEEE*, vol. 16, no. 1, pp. 303–336, 2014, doi: 10.1109/SURV.2013.052213.00046.

[2] N. McKeown, "How SDN will shape networking," *ONS 2011*, 2011. https://www.youtube.com/watch?v=c9-K5O_qYgA (accessed Jan. 20, 2018).

[3] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, 2013, doi: 10.1109/MCOM.2013.6461195.

[4] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN Security: A Survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, 2013, pp. 1–7, doi: 10.1109/SDN4FNS.2013.6702553.

[5] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *Commun. Surv. Tutorials, IEEE*, vol. PP, no. 99, p. 1, 2015, doi: 10.1109/COMST.2015.2453114.

[6] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, 2014, pp. 167–171, doi: 10.1109/ICACT.2014.6778942.

[7] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "A survey and classification of the security anomaly detection mechanisms in software defined networks," *Cluster Comput.*, pp. 1–19, Sep. 2020,

doi: 10.1007/s10586-020-03184-1.

[8] Y. Hande and A. Muddana, “A survey on intrusion detection system for software defined networks (SDN),” *Int. J. Bus. Data Commun. Netw.*, vol. 16, no. 1, pp. 28–47, 2020, doi: 10.4018/IJBDCN.2020010103.

[9] Y. Zhang, “An adaptive flow counting method for anomaly detection in SDN,” *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, Santa Barbara, California, USA, pp. 25–30, 2013, doi: 10.1145/2535372.2535411.

[10] T. Ha *et al.*, “Suspicious traffic sampling for intrusion detection in software-defined networks,” *Comput. Networks*, vol. 109, Part, pp. 172–182, 2016, doi: http://dx.doi.org/10.1016/j.comnet.2016.05.019.

[11] B. R. Granby, B. Askwith, and A. K. Marnerides, “SDN-PANDA: Software-Defined Network Platform for ANomaly Detection Applications,” in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, 2015, pp. 463–466, doi: 10.1109/ICNP.2015.58.

[12] S. Hommes, R. State, and T. Engel, “Implications and detection of DoS attacks in OpenFlow-based networks,” *2014 IEEE Glob. Commun. Conf. GLOBECOM 2014*, pp. 537–543, 2014, doi: 10.1109/GLOCOM.2014.7036863.

[13] L. F. Carvalho, G. Fernandes, J. J. P. C. Rodrigues, L. S. Mendes, and M. L. Proenca, “A novel anomaly detection system to assist network management in SDN environment,” *IEEE Int. Conf. Commun.*, 2017, doi: 10.1109/ICC.2017.7997214.

[14] D. He, S. Chan, X. Ni, and M. Guizani, “Software-Defined-Networking-Enabled Traffic Anomaly Detection and Mitigation,” *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1890–1898, Dec. 2017, doi: 10.1109/JIOT.2017.2694702.

[15] L. F. Carvalho, T. Abrão, L. de S. Mendes, and M. L. Proença, “An ecosystem for anomaly detection and mitigation in software-defined networking,” *Expert Syst. Appl.*, vol. 104, pp. 121–133, Aug. 2018, doi: 10.1016/J.ESWA.2018.03.027.

[16] H. Peng, Z. Sun, X. Zhao, S. Tan, and Z. Sun, “A Detection Method for Anomaly Flow in Software Defined Network,” *IEEE Access*, 2018, doi: 10.1109/ACCESS.2018.2839684.

[17] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6961 LNCS, pp. 161–180, doi: 10.1007/978-3-642-23644-0_9.

[18] S. E. Schechter, J. Jung, and A. W. Berger, “Fast detection of scanning worm infections,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3224, pp. 59–81, 2004, doi: 10.1007/978-3-540-30143-1_4.

[19] J. Twycross and M. M. Williamson, “Implementing and Testing a Virus Throttle,” in *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*, 2003, p. 20.

[20] M. M. Williamson, “Throttling viruses: Restricting propagation to defeat malicious mobile code,” in *Proceedings - Annual Computer Security Applications Conference, ACSAC*, 2002, vol. 2002-January, pp. 61–68, doi: 10.1109/CSAC.2002.1176279.

[21] M. V. Mahoney, “Network traffic anomaly detection based on packet bytes,” in *Proceedings of the 2003 ACM symposium on Applied computing - SAC ’03*, 2003, p. 346, doi: 10.1145/952532.952601.

[22] R. T. Kokila, S. Thamarai Selvi, and K. Govindarajan, “DDoS detection and analysis in SDN-based environment using support vector machine classifier,” in *6th International Conference on Advanced Computing, ICoAC 2014*, Aug. 2015, pp. 205–210, doi: 10.1109/ICoAC.2014.7229711.

[23] R. Sathya and R. Thangarajan, “Efficient anomaly detection and mitigation in software defined networking environment,” in *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, 2015, pp. 479–484, doi: 10.1109/ECS.2015.7124952.

[24] C. Yang, “Anomaly network traffic detection algorithm based on information entropy measurement under the cloud computing environment,” *Cluster Comput.*, vol. 22, no. 4, pp. 8309–8317, Jul. 2019, doi: 10.1007/s10586-018-1755-5.

[25] R. Wang, Z. Jia, and L. Ju, “An entropy-based distributed DDoS detection mechanism in software-defined networking,” in *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, Dec. 2015, vol. 1, pp. 310–317, doi: 10.1109/Trustcom.2015.389.

[26] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, “Combining Open Flow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments,” *Comput. Networks*, vol. 62, pp. 122–136, 2014, doi: 10.1016/j.bjp.2013.10.014.

[27] N. Visible and H.- Packard, “Traffic Monitoring using sFlow,” *Analysis*, 2003. http://www.sflow.org/sFlowOverview.pdf.

[28] J. Francois and O. Festor, “Anomaly traceback using software defined networking,” in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014, pp. 203–208, doi: 10.1109/WIFS.2014.7084328.

[29] L. Deng and D. Yu, “Deep learning: Methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4. Now Publishers Inc, pp. 197–387, 2013, doi: 10.1561/2000000039.

[30] S. K. Dey and M. M. Rahman, “Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method,” in *4th International Conference on Electrical Engineering and Information and Communication Technology, iCEEiCT 2018*, Jan. 2019, pp. 630–635, doi: 10.1109/CEEICT.2018.8628069.

[31] Q. Niyaz, W. Sun, and A. Y. Javaid, “A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN),” *arXiv Prepr. arXiv1611.07400*, 2016.

[32] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for Network Intrusion Detection in Software Defined Networking,” in *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, Dec. 2016, pp. 258–263, doi: 10.1109/WINCOM.2016.7777224.

[33] S. Garg, N. Kumar, J. J. P. C. Rodrigues, and J. J. P. C. Rodrigues, “Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective,” *IEEE Trans. Multimed.*, vol. 21, no. 3, pp. 566–578, Mar. 2019, doi: 10.1109/TMM.2019.2893549.

[34] J. Li, Z. Zhao, and R. Li, “Machine learning-based IDS for softwaredefined 5G network,” *IET Networks*, vol. 7, no. 2, pp. 53–60, Mar. 2018, doi: 10.1049/iet-net.2017.0212.

[35] A. Santos Da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, “ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN,” *Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp.*, no. Noms, pp. 27–35, 2016, doi: 10.1109/NOMS.2016.7502793.

[36] C. Pang, Y. Jiang, and Q. Li, “FADE: Detecting forwarding anomaly in software-defined networks,” Jul. 2016, doi: 10.1109/ICC.2016.7510990.

[37] Y. Cui *et al.*, “SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks,” *J. Netw. Comput. Appl.*, vol. 68, pp. 65–79, 2016, doi: http://dx.doi.org/10.1016/j.jnca.2016.04.005.

[38] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in *Proceedings - Conference on Local Computer Networks, LCN*, 2010, pp. 408–415, doi: 10.1109/LCN.2010.5735752.

[39] “Open Networking Foundation.” https://www.opennetworking.org/.

[40] ONF and O. N. Foundation, “Software-Defined Networking: The New Norm for Networks,” 2012. doi: citeulike-article-id:12475417.

[41] Y. Jarraya, T. Madi, and M. Debbabi, “A Survey and a Layered Taxonomy of Software-Defined Networking,” *Commun. Surv. Tutorials, IEEE*, vol. PP, no. 99, p. 1, 2014, doi: 10.1109/COMST.2014.2320094.

[42] N. McKeown *et al.*, “OpenFlow: Enabling Innovation in Campus Networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008, doi: 10.1145/1355734.1355746.

[43] L. Deng and D. Yu, “Deep learning: methods and applications,” *Found. trends signal Process.*, vol. 7, no.

3–4, pp. 197–387, 2014.

[44] R. Hecht-Nielsen, “Replicator neural networks for universal optimal source coding,” *Science (80-.).*, vol. 269, no. 5232, pp. 1860–1863, 1995.

[45] S. Hawkins, H. He, G. Williams, and R. Baxter, “Outlier detection using replicator neural networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002, vol. 2454 LNCS, pp. 170–180, doi: 10.1007/3-540-46145-0_17.

[46] C. G. Cordero, S. Hauke, M. Muhlhauser, and M. Fischer, “Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks,” in *2016 14th Annual Conference on Privacy, Security and Trust, PST 2016*, 2016, pp. 317–324, doi: 10.1109/PST.2016.7906980.

[47] H. A. Dau, V. Ciesielski, and A. Song, “Anomaly detection using replicator neural networks trained on examples of one class,” in *Asia-Pacific Conference on Simulated Evolution and Learning*, 2014, pp. 311–322.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[49] L. Tóth and G. Gosztolya, “Replicator neural networks for outlier modeling in segmental speech recognition,” in *International Symposium on Neural Networks*, 2004, pp. 996–1001.

[50] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[51] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection.” Accessed: May 04, 2019. [Online]. Available: https://arxiv.org/pdf/1607.00148.pdf.

[52] O. Gu, P. Fogla, D. Dagon, W. Lee, and B. Škorić, “Measuring intrusion detection capability: An information-theoretic approach,” in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS ’06*, 2006, vol. 2006, pp. 90–101, doi: 10.1145/1128817.1128834.

[53] Y. Xin *et al.*, “Machine Learning and Deep Learning Methods for Cybersecurity,” *IEEE Access*, p. 1, 2018, doi: 10.1109/ACCESS.2018.2836950.

[54] K. A. Spackman, “SIGNAL DETECTION THEORY: VALUABLE TOOLS FOR EVALUATING INDUCTIVE LEARNING,” in *Proceedings of the Sixth International Workshop on Machine Learning*, Elsevier, 1989, pp. 160–163.

[55] T. Fawcett, “ROC Graphs: Notes and Practical Considerations for Researchers,” *HP Labs Tech Rep. HPL-2003-4*, pp. 1–38, 2004, doi: 10.1.1.10.9777.

[56] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley, 2013.

[57] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, “PayLess: A low cost network monitoring framework for Software Defined Networks,” in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9, doi: 10.1109/NOMS.2014.6838227.

[58] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanović, “Flow-based anomaly intrusion detection system using two neural network stages,” *Comput. Sci. Inf. Syst.*, vol. 11, no. 2, pp. 601–622, 2014, doi: 10.2298/CSIS130415035A.

[59] Floodlight, “Floodlight OpenFlow Controller.” http://www.projectfloodlight.org/floodlight/.

[60] “Open vSwitch.” http://openvswitch.org/.

[61] Mininet Team, “Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet.” http://mininet.org/.

[62] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012, doi: http://dx.doi.org/10.1016/j.cose.2011.12.012.

[63] “Tcpreplay.” http://tcpreplay.synfin.net/.

[64] “Hping - Active Network Security Tool.” http://www.hping.org/ (accessed Nov. 16, 2020).

[65] S. Zavrak and M. Iskefiyeli, “Anomaly-Based Intrusion Detection from Network Flow Features Using Variational Autoencoder,” *IEEE Access*, vol. 8, pp. 108346–108358, 2020, doi: 10.1109/ACCESS.2020.3001350.

[66] V. L. Cao, M. Nicolau, and J. McDermott, “Learning Neural Representations for Network Anomaly Detection,” *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074–3087, Aug. 2019, doi: 10.1109/TCYB.2018.2838668.

[67] J. Patterson and A. Gibson, *Deep Learning: A Practitioner’s Approach*. O’Reilly Media, 2017.

[68] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” 2013. Accessed: Jan. 30, 2019. [Online]. Available: http://www.cs.toronto.edu/~fritz/absps/momentum.pdf.

[69] “KDD-OpenSource/DeepADoTS: Repository of the paper ‘A Systematic Evaluation of Deep Anomaly Detection Methods for Time Series’.” https://github.com/KDD-OpenSource/DeepADoTS (accessed Mar. 29, 2021).

[70] “PyTorch.” https://pytorch.org/ (accessed Nov. 16, 2020).

[71] S. Aksoy and R. M. Haralick, “Feature normalization and likelihood-based similarity measures for image retrieval,” *Pattern Recognit. Lett.*, vol. 22, no. 5, pp. 563–582, 2001.

[72] W. J. Krzanowski and D. J. Hand, *ROC Curves for Continuous Data*. CRC Press, 2009.

[73] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks,” in *2018 4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018*, Sep. 2018, pp. 462–469, doi: 10.1109/NETSOFT.2018.8460090.

[74] A. Abubakar and B. Pranggono, “Machine learning based intrusion detection system for software defined networks,” in *Proceedings - 2017 7th International Conference on Emerging Security Technologies, EST 2017*, 2017, pp. 138–143, doi: 10.1109/EST.2017.8090413.

[75] P. Wang, K. M. Chao, H. C. Lin, W. H. Lin, and C. C. Lo, “An Efficient Flow Control Approach for SDN-Based Network Threat Detection and Migration Using Support Vector Machine,” in *2016 IEEE 13th International Conference on e-Business Engineering (ICEBE)*, 2016, pp. 56–63, doi: 10.1109/ICEBE.2016.020.