Journal of Network and Computer Applications 191 (2021) 103160

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique

Mahmoud Said ElSayed [a,*], Nhien-An Le-Khac [a], Marwan Ali Albahar [b], Anca Jurcut [a]

[a] School of Computer Science, University College Dublin, Belfield, Dublin, Ireland
[b] School of Computer Science, Umm Al-Qura University, Mecca, Saudi Arabia

ARTICLE INFO

Keywords:
Convolutional neural network (CNN)
Deep Learning (DL)
InSDN dataset
Intrusion detection system (IDS)
Machine learning
Overfitting
SDN
SD-Reg

ABSTRACT

Software-defined networking (SDN) is a new networking paradigm that separates the controller from the network devices i.e. routers and switches. The centralized architecture of the SDN facilitates the overall network management and addresses the requirement of current data centers. While there are high benefits offered by the SDN architecture, the risk of new attacks is a critical problem and can prevent the wide adoption of SDNs. The SDN controller is a crucial element, and it is an attractive target for the intruders. In case the attacker successfully accessed the SDN controller, it can route the traffic based on its own requirements, causing severe damage to the entire network. The network intrusion detection systems (NIDSs) are important tools to detect and secure the network environment from malicious activities and anomalous attacks. Deep Learning (DL) has recently shown desirable results in a variety of problems, such as text, speech, and image applications, etc.

While several related works deployed DL for NIDSs, most of these approaches ignore the influence of the overfitting problem during the implementation of DL algorithms. As a result, it can impact the robustness of the anomaly detection system and lead to poor model performance for zero-day attacks. In this work, we propose a new hybrid DL approach based on the convolutional neural network (CNN) to classify the flow traffic into normal or attack classes. A new regularizer method, namely SD-Reg, which is based on the standard deviation of the weight matrix, has been used to address the problem of overfitting and to improve the capability of NIDSs in detection of unseen intrusion events. The evaluation results indicate that the SD-Reg outperforms the previous regularizer methods. In addition, the proposed hybrid technique gives a higher performance in all the evaluation metrics compared to the single DL models. Several datasets, including the InSDN – the most recent dataset for SDN – are used to train and evaluate the performance of all techniques. Furthermore, we suggest a lightweight NIDS by training the CNN-based models using a less number of features without causing a significant drop in the model performance.

1. Introduction

SDN is an emerging computer network paradigm to mitigate the current challenges of conventional networks, which suffer from inflexibility in management and higher maintenance cost. The new emerging paradigm provides a global view of the network system and facilitates centralized management through the programmable control plane, making the network more flexible to deploy different functions (Jahromi et al., 2018; Elsayed et al., 2021). Several use cases in the networking world, such as traffic engineering, network monitoring, quality of service (QoS) and datacenter networking have applied SDN. The flexible nature of SDN accelerates innovation research and enhances security measures such as threat detection and prevention compared to the conventional networks.

Although of the high benefits of the SDN, it has certain security challenges that need to be addressed for broad adoption of the new paradigm. For example, the security breach in the conventional networks has limited damage for only a small part of the network i.e. probably for the same network vendor, while any attack against the SDN controller may lead to severe consequences on the whole network. In case the attacker succeeds in bringing the controller down, the network might be exposed to severe crashes. Also, the attacker can flood the network with the most dangerous attacks in SDN such as Denial of Service (DoS) or Distributed Denial of Service (DDoS)

* Corresponding author.
E-mail addresses: mahmoud.abdallah@ucdconnect.ie (M.S. ElSayed), an.lekhac@ucd.ie (N. Le-Khac), mabahar@uqu.edu.sa (M.A. Albahar), anca.jurcut@ucd.ie (A. Jurcut).

https://doi.org/10.1016/j.jnca.2021.103160
Received 3 March 2021; Received in revised form 25 May 2021; Accepted 28 June 2021
Available online 9 July 2021

attacks (Elsayed et al., 2020a). Thus, the legitimate requests might be denied since the channel bandwidth and the network resources are heavily consumed (Elsayed et al., 2020a). Kreutz et al. (2014) represented seven attack vectors that can strike SDNs. Among these attacks, three vectors are specific for SDN networks, including SDN controller vulnerabilities, the attacks on the control plane communication, as long with the administrator host. Another study (Klöti et al., 2013) employed stride and attack tree approaches to represent security analysis and modeling methodologies in the SDN.

Network intrusion detection systems (NIDSs) are standard security solutions to detect various intrusions activities that can be observed inside an organizational network (Kumar, 2007). The basic functionality of the NIDS is to observe and to analyze the incoming and outgoing network traffic and to generate an alarm, reporting suspicious activity. There are two main classes of NIDSs based on the detection approach: signature-based NIDS and anomaly-based intrusion detection system (Khraisat et al., 2019). In the signature-based NIDS eg. Snort, the signature of attacks is stored in the acknowledge database. In case the observed traffic is matched with the stored signature, an alarm is generated, referring to the detected attack. While this kind of intrusion detection is widely applied for commercial products, since it has a low false alarm and provides high detection accuracy; it is only effective in detecting the known attacks. However, any change in the attack signature, even small amendments from the attacker, can easily bypass the functionality of such techniques without being noticed. In addition, the signature database should be periodically updated to include the new signature of the discovered attacks. Adding new signatures can increase the size of the database and slow the process of analyzing and monitoring. On the other hand, the anomaly-based solutions are more attractive for the research community since they are able to observe any deviation from the normal traffic pattern. Although anomaly-based detection techniques are theoretically suited to identify the unknown attacks, they experience insufficient analysis capability and low detection rate. Thus, the development of the anomaly-based IDS is a primary research direction since the security challenges are among the most critical issues facing SDNs. The statistical and ML techniques are widely applied for anomaly-based solutions (Zarpelão et al., 2017).

Several ML approaches such as Naive–Bayes (NB), decision tree (DT), and support vector machine (SVM) have been applied to develop effective anomaly IDSs. However, applying these approaches is based on manually extracted features, which may cause loss of the original flow information. Besides, they have a high computational cost and can output high false positive alarms, which reduce the implementation of these approaches in real-time in actual situations.

In recent years, the DL-based approaches, a subset of ML paradigm, achieved significant results when applied in many computer vision scenarios, such as image captioning, translating natural language, speech recognition, and many others (Pouyanfar et al., 2018). The DL techniques have the significant ability to automatically extract the discriminatory features from high-dimensional data while avoiding the necessity of feature engineering usage. Several studies have been applied DL for intrusion detection systems and achieved better performance compared to traditional ML techniques (Xin et al., 2018). Motivated by this fact, this paper proposes a novel hybrid learning approach to enhance the security under the SDN context. The proposed approach combining the convolutional neural network (CNN) algorithm with several ML algorithms, including random forests (RF), k-nearest neighbors (KNN), and SVM. The developed solution uses SD-Reg, a new regularization technique (Albahar, 2019) based on the standard deviation method to address the problem of overfitting.

The contributions of this paper include the following points:

- Investigates ML/DL solutions in literature for NIDSs and discusses the gap in this research.
- Proposes a novel hybrid models that is based on a new regularizer technique, namely SD-Reg and the CNN. The SD-Reg technique uses the standard deviation to overcome the overfitting problem of the classifier models. The models based on the SD-Reg achieved the best results compared with the existing methods i.e. L1 and L2.
- Several experiments are performed to verify the performance of the proposed hybrid models for binary and multi-class classifications. In addition, a small set of features are utilized to demonstrate how DL models can perform using only a few features. All experiments are tested using the InSDN dataset, which was generated to address the leakage of datasets in SDN networks (Elsayed et al., 2020c).
- We demonstrate how the DL approaches can perform with new attacks that were never seen due to the training phase. The achieved results are compared against various ML and DL algorithms such as logistic regression (LR), NB, SVM, RF, KNN, DT and long short term memory (LSTM).
- Validates the performance of the proposed DL approaches on two different benchmark datasets, namely, the CSE-CIC-IDS2018 (Sharafaldin et al., 2018) and the UNSW-NB15 (Moustafa and Slay, 2015).
- The effectiveness and performance of proposed DL approaches are evaluated using various evaluation metrics i.e. accuracy, precision, recall, F1-score, and area under curve. The reported results indicate that the hybrid models have higher performance in terms of all evaluation metrics than the classical ML algorithms.

The rest of this paper is organized as follows: Section 2 provides a brief background about the CNN structure and popular ML algorithms. Section 3 provides an in-depth view of the existing promising countermeasures that have been produced to monitor and detect threats. The regularization techniques and the implementation details of the proposed method using CNN are reported in Section 4. Experiments and the dataset analysis are described in Section 5. Experimental results and analysis are discussed in Section 6. Section 7 gives a discussion and details the article limitation. The paper conclusion is discussed in the last section.

2. Background theory

This section provides the theoretical background for several classifiers, which are applied to our classification problem.

2.1. Convolutional neural networks (CNN)

The CNN is a specialized type of neural network (NN) model, widely used for computer vision. It can effectively address the parameter explosion problem of traditional NN by allowing the parameters sharing through each layer. The concept of the CNN is composed of three core layers, namely Convolution, pooling and fully connected layers (Yamashita et al., 2018). 'Convolution' refers to a linear operation, resulting from the multiplication of a set of weights called filter or kernel with the input. The filter is smaller than the input data to allow the same filter to be multiplied by the input array several times at different regions of the input. The filter matrix is moved over the original input by a side of window called 'stride'. For example, when the stride is 1 then we move the filters one pixel at a time. However, using a large value of the stride will produce smaller feature maps. The result from multiplying the filter with the previous layer one time is a single value. Dragging the filter multiple times across the entire previous layer produces a new array with two dimensions, which represents filtering of the input i.e. feature map. The units within a feature map share the same weight matrix. The formula used to calculate the feature map values is presented in the equation:

$$s = \theta(x \otimes w + b) \tag{1}$$

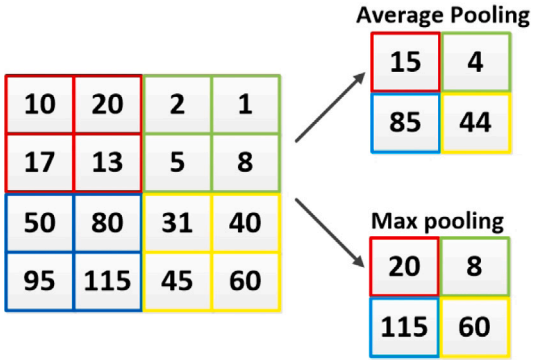


Fig. 1. Example of the different methods of pooling layers.

where s is the feature map, x is the input data, b is the bias term, w is the weight of the kernel function, and θ is the non-linear activation function used. The Relue function is wildly used in CNN to set all negative values in the feature map by zero. Furthermore, it increases the degree of non-linearity inside the convolutional layers, since the convolution is a linear operation and most of the real-world data would be non-linear.

The CNN can have several convolutional layers. The first layer is effectively used to capture the low-level features like edges or corners, while the high-level features are obtained within the next layers. As a result, using several convolutional layers has a wholesome understanding of the input data. However, the output dimensional of the convolutional layers is less compared to the input, since the image size is reduced every time the convolutional is performed. Subsequently, the input image disappears completely after only a limited number of times. To solve this problem, the padding technique can be used to pad the image with an additional border. There are two types of convolution: Same and Valid. The same padding uses a border around the image, so the input and the output images have the same size. The padding size should meet the following equation formula:

$$\rho = \frac{f-1}{2} \tag{2}$$

where ρ is the padding size, and f is the filter size. In valid, the original image is used i.e. the zero pixel padding around the input matrix is not added. The CNN produces the pooling layer concept to reduce the spatial size of the convolved feature. There are two different types of pooling: max pooling and average pooling. In max pooling, the largest value is returned from the portion of the input image covered by the kernel filter, while the average of all values is returned in the case of the average pooling. The max pooling is widely preferred and applied more than the average method due to its ability to perform de-noising along with dimensionality reduction. Using the pooling techniques can reduce the dimensionality of the extracted feature maps while preserving information. An example of the pooling technique is depicted in Fig. 1. After the previous process, the output is going to the flatten layer and then passed through a fully connected layer for classification purposes. Nowadays, there are several architectures available to improve the performance of CNNs such as LeNet (LeCun et al., 2015), AlexNet (Iandola et al., 2016), VGGNet (Simonyan and Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), ResNet (Xie et al., 2017) and ZFNet (Zeiler and Fergus, 2014).

2.2. SVM

SVM is a type of supervised ML algorithm applied widely for classification and regression. The SVM works to find the optimal separate hyper-plane by projecting data into a high-dimensional feature space using the kernel trick. The SVM is designed for binary classification but also can be employed for multiclass classification problems.

2.3. K-nearest neighbors (KNN)

The KNN is one of the most simple algorithms applied widely for regression and prediction. It seeks to classify the data point based on the distance between them by assuming that the similar samples are close to each other. The Euclidean distance is one of the popular and familiar choices to calculate the distance between data samples. However, the KNN is an impractical choice and is becoming significantly slower when used in large-scale data. Whereas, increasing the data volume size dramatically decreases the distance between the data points. Hence, using DL in the pre-trained phase can solve the problem of KNN dimensionality.

2.4. Random forest

In the individual decision trees (DTs), the feature space is partitioned into smaller and smaller sub-spaces to best find the target value. It starts with a single root, which then branches off into a number of solutions, just like a tree, by applying a sequence of simple rules. These rules are estimated by identifying the relationship between the input and the output. The splitting process will continue until any matching rule is presented, or no further gain can be made. The individual decision trees uses the whole dataset for the model training. However, the DT is very prone to the overfitting problem, where the model performs well during the training but is not so flexible for making predictions on unseen samples. To overcome this problem, the Random Forest (RF) combines multiple individual decision trees together that operate as an ensemble (Oshiro et al., 2012). Each data point in RF is randomly chosen from the whole dataset, and a data point can be picked more than once. By using different samples of data to train each individual tree, the individual tree overfitting can be solved. The class prediction in the RF is building from each individual tree separately, and the most vote class becomes the model prediction. The RF outperforms the individual trees, since the higher number of trees is less prone to the individual errors. The group of trees can find the correct direction even some trees of the group may be wrong in their calculation.

3. Literature review

NIDS is one of the powerful security defenses to deter and mitigate the computer network security against malicious attacks. Several traditional methods have been proposed to detect malicious behavior in networks, such as threshold analysis methods (Xu and Mueller, 2018), statistical methods (Verma and Ranga, 2018), and signature methods (Wang et al., 2018). However, these techniques perform very poor when the amount of the network data is significantly huge. Unfortunately, these methods rely on the security researchers experience to quantify the malicious traffic behavior based on the past observation. For example, selecting a sub-optimal threshold value for any network system may not be reliable or being easy to identify. It often takes a long time to monitor network traffic in real time in order to find out the optimal threshold value. In addition, the network states are not consistent all the time. The traffic size may be very busy or idle, depending on the network conditions. Thus, the nodes could observe variations in packet arrival rate. Therefore, the threshold-based methods are not preferred for detection and classification applications. Additionally, these methods can substantially lead to higher false-alarm instances or the miss-detection.

The last decade has witnessed a significant increasing in the use of ML and DL techniques to address the requirements for developing powerful IDSs.

In the following subsections, we investigate the most recent approaches that employed ML and DL to provide a quick response against malicious attack traffic in different network environments.

3.1. IDSs based on ML

Li et al. (2018) proposed an anomaly detection based technique to protect the SDN network. The presented model composites of two steps. In the first step, the Bat algorithm (BA), with swarm division and binary differential mutation, is leveraged to select important attributes and reduces the number of features from high dimensional data. In the second stage, RF is employed using the selected features from the first stage to classify the network traffic into several classes.

An adaptive ensemble learning model (Gao et al., 2019) was proposed by setting up multiple DT with adaptive voting algorithm. The NSL-KDD benchmark dataset was used to verify the proposed model. From the experiment results, the adaptive ensemble ML model has better results than the single algorithm. The prediction accuracy is significantly improved and the obtained accuracy reached 84.23% and 85.2% when using MultiTree and Ensemble Voting, respectively.

Yulianto et al. (2019) used the AdaBoost algorithm for anomaly-based system. Two feature selection algorithms i.e. Ensemble Feature Selection (EFS) and Principal Component Analysis (PCA) are utilized to select typical features from the CICIDS 2017 Dataset. The reported results showed that the integration of the AdaBoost with EFS gives higher performance than AdaBoost with PCA.

Jan et al. (2019) applied the SVM algorithm to develop a light-weight attack detection strategy against malicious attacks in IoT networks. The model is trained using three features only derived from the packet arrival rate attribute. The three extracted features are obtained by calculating the mean, median and maximum values of packet arrival rate attribute. Although the limited number of used features performs efficiently in discriminating an intrusion in the IoT network, however, considering only one attribute for training the intrusion system, it makes the method not a promising solution.

Abubakar and Pranggono (2017) proposed two IDSs to protect the SDN network from malicious traffic. The signature-based Snort IDS is implemented for known attack detection, while the flow-based anomaly detection is used as a second IDS to provide a scalable threat detection in the network architecture. The anomaly detection model, which is based on pattern recognition of NNs is evaluated on the NSL-KDD benchmark dataset. However, deploying two IDSs can add an additional layer of complexity to the network and increase the computational time, which reflects on the traffic delay in the network.

Santos et al. (2020) examined the performance of four various ML-Algorithms, namely SVM, RF, DT and Multiple Layer Perceptron (MLP), against DDoS attacks under the SDN context. The Scapy tool is employed to generate benign and malicious traffic. The Mininet with POX controller are used in order to simulate the SDN network. Although this work avoided the leakage availability of the dataset in the SDN network, the generated traffic is only limited for DDoS attacks, without considering the various attacks that can occur in the SDN network. Furthermore, the network traffic losses to diversity, since only Scapy tool was employed to generate all data traffic.

However, most of these approaches can be considered shallow learning strategies and cannot meet the requirements to detect complex malicious attacks. For example, using SVM when the processing data has a huge amount of samples, might consume considerable computational resources. On the other hand, applying the DT algorithm on data that contains a lot of noise can easily lead to model overfitting problem and exhibit low classification accuracy. Moreover, the effectiveness of these techniques in the classification of IDS relies heavily on feature extraction methods. However, the feature selection methods are not an easy task and always need exporters. The features that can be used for one attack category may not be suitable for other categories, since the attack scenarios are continuously changing and evolving (Elsayed et al., 2019). Moreover, developing discriminative features methods is useless when only normal data is available. On the other hand, the ML techniques provide high performance when the labeled data has a small sample size. Such techniques suffer from several limitations and are not significant when applied on huge traffic data.

In recent years, several DL approaches have been applied for intrusion detection systems. The DL has a more robust ability to learn the deep structure from raw data and obtain the features automatically, instead of using manual feature selection and extraction.

3.2. IDSs based on DL

In Elsayed et al. (2020b), authors employed DL to solve the problem of the minority attacks in imbalanced datasets. The proposed model integrated the autoencoder with Long Short Term Memory (LSTM) in unsupervised learning. The model was trained using only normal data and then it was applied in testing data, which composites from normal and malicious traffic. While the training error is very low, due to the training phase, since only the samples from the normal data are used; it is relativity high for the malicious traffic. The error is utilized as a threshold to separate between normal and malicious boundaries. Although the proposed model provided effective results for the NSL-KDD dataset, it failed to give a good performance with recent attack datasets. This is because the modern datasets contain sophisticated attacks, which are not easy to be separated using a simple threshold.

Authors extended the aforementioned work and integrated the One-class SVM (OC-SVM) with the LSTM-Autoencoder, to overcome the limitation of a simple threshold (Said Elsayed et al., 2020). The compressed features (i.e. latent) that are extracted from the LSTM-Autoencoder are used to train another OC-SVM algorithm in unsupervised learning, in order to detect anomalies in the testing data. The results showed that by integrating the OC-SVM with DL approaches, it can significantly improve the anomaly detection rate when compared to the single OC-SVM algorithm.

Vigneswaran et al. (2018) analyzed the performance of the deep neural network (DNN) model in comparison to classical ML techniques. All experiments have been applied on KDDCup-'99'. The experiment results showed that the DNN with three hidden layers outperformed all other classical ML algorithms.

Tang et al. (2018) proposed a DL approach using Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) against DDoS attacks in SDN networks. The learned model achieved accuracy reached 89% using only six raw features extracted from NSL-KDD dataset.

Similar to Tang et al. (2018), Isa and Mhamdi (2020) introduced a hybrid DL approach based on auto-encoding and RF algorithm to address the problem of DDoS attacks in SDNs. The reported accuracy of the proposed approach reached to 98.4% using the NSL-KDD benchmark dataset.

Kim et al. (2017) used LSTM to effectively train the intrusion detection model using KDDCup-'99'. The model was tested using six different optimizers i.e. RMSprop, Adagrad, Adadelta, Adam, and Adamax. For the experiment results, the Nadam optimizer enhanced the intrusion detection accuracy and outperformed the other optimizers.

Another study (Farahnakian and Heikkonen, 2018) applied the Deep Auto-Encoder (DAE) for intrusion detection to identify malicious network traffic. The Softmax classifier is used in the last hidden layer to classify the input data into normal or attack categories. The greedy layer-wise fashion was used to train the DL model to avoid the overfitting and local optima. The DAE model was trained in unsupervised manner to learn the representation of important features from the trained dataset, while another supervised learning strategy was deployed to perform fine-tunining in all layers through back-propagation algorithm.

Javaid et al. (2016) employed the sparse autoencoder with Softmax classifier against intrusion attacks. The DL approach is trained using the NSL-KDD dataset. The good feature representation is extracted in the first stage using the sparesencoder in unsupervised learning and the Softmax layer is used for the classification task.

An effective DL approach (Al-Qatf et al., 2018) integrated the sparse autoencoder (SAE) with SVM for intrusion detection on the NSL-KDD dataset. The sparse autoencoder mechanism processed feature extraction and dimensionality reduction in unsupervised learning, while SVM is used for classification process to enhance the detection capability of the model. The results showed that the hybrid model can effectively improve the accuracy rate and reduce the training and testing times simultaneously.

Although the fully connected DL approaches can potentially offer a better solution for the implementation of IDSs, the traditional neural networks are inspired by a huge number of training parameters (i.e. the adjacent layer units are fully connected together). However, the extensive parameters can limit the size of neural network layers and slow down the training process. Instead, CNNs can solve effectively the problem of the explosion of neural network parameters through strategies of shared weights and sparse connectivity (Gu et al., 2018). The CNN is a special kind of feed-forward artificial neural networks, combining the convolution and pooling operations together to extract the feature vectors from the input data and improve classification accuracy. However, only a limited numbers of works have used CNNs for IDSs.

A new IDS based on CNN (Liu et al., 2017) was proposed on the KDDCup-'99' dataset. It used the same LeNet-5 CNN architecture (Depren et al., 2005), which contains two convolutional layers, two maxpool layers, two full-connection layers, and 1 classification layer. Different 10 test datasets from KDDCup are used during the evolution phase, where each test data has 5000 randomly selected examples. The average detection rate and average false alarm are 0.976614 and 0.099958, respectively. However, the proposed approach fails to detect any sample of U2R class. Besides, the average accuracy of Probe and R2L attacks are 56.26 and 61.47, respectively.

Lin et al. (2018) used CNN based on Le-Net-5 model with Softmax function to develop a new classifier-based learning model for network anomaly detection. The information gain (IG) technique is used in the first stage for feature reduction purposes in order to select the top 32 features instead of 41. The model is employed to classify the six major types of threats i.e. smurf, Neptune, normal, satan, ipsweep, portsweep. The authors successfully achieved an accuracy of 99.65% using 8 cross-validation technique. However, using IG as an initial step for feature reduction before CNN can increase the complexity of the DL approach, it can be unnecessary, since the DL approach can automatically extract the important attributes from the input data without any feature engineering approach.

Xiao et al. (2019) introduced a new DL model based on the CNN algorithm using the KDDCup-'99' dataset. The reduction techniques i.e. PCA and autoencoder are used in the first stage in order to achieve the requirements of the CNN. The input vector dimensions are reduced from (1×122 vector) to 1×121 or 1×100 dimension reduction vectors. The network feature vectors are converted to image format with 11×11 or 10×10 matrices, and then the transformed 2-dimensional matrix is passed to the CNN input layer. The CNN model i.e. CNN-IDS, which is based on Lenet-5 typical model, is proposed to extract and analyze the characteristics of the network traffic. The authors achieved an overall accuracy of 94.0%, but they failed to achieve a reasonable performance for U2R and R2L attacks (the detection rates of U2R and R2L are 20.61% and 18.96%, respectively). However, the common point in the aforementioned works is that they all applied CNN alone; there are no proposed methods to combine the use of CNNs and ML techniques for a better overall classification.

In particular, the computational time for CNN is relatively low compared to other classification algorithms. However, most of the existing studies evaluated their models using outdated datasets such as KDDCup-'99' and NSL KDD that are not suitable for modern IDSs. Training the IDSs using old datasets can cause a significant issue, as the intrusion attack types are constantly changing and are becoming more sophisticated and not easy to identify. Furthermore, the available datasets were generated based on the conventional networks, not from SDN environments. Whereas, the specific characteristics of the SDN pose an adversary to launch attacks in many ways in order to disrupt the network system. Moreover, the used protocols in SDN are different from those used in conventional networks, e.g., Openflow protocol. The new protocol introduces new weaknesses, resulting in new challenges for IDSs in the SDN. Hence, we prompt studies on evaluating the DL models using the up-to-date SDN datasets to implement a more realistic IDS.

This paper developed an improved behavior-based classifier learning by training CNN with a new regularization technique to improve the classification detection rate of cyber threats and attacks. In addition, different ML algorithms i.e. SVM, DT and RF are used for the classification process instead of the Softmax function to improve the classification accuracy and to minimize the computational complexity.

4. Adopted approach

This section describes the SD-Reg techniques used in our proposed model in comparison with traditional regularization methods. Further, we present the architecture and detailed explanation of the learning model, considering the used parameters in each layer.

4.1. Proposed regularizer

In this section, we discuss in-depth the regularization techniques that can help with the overfitting problem and to improve the model interpretability.

4.1.1. Regularization: A method of controlling the model from complexity

A key issue in ML is how to build an algorithm which works well on both training data and new inputs. Many ML techniques are deliberately designed to reduce the test error at the cost of increased training error. The regularization process is applied to reduce the model generalization error. Regularization seeks to penalize the learning model by performing well on unseen data to begin generalization. Different types of regularizers put the predictive model on penalty terms. L1 and L2 regularizers are the most prominently used regularization techniques (Ng, 2004). Below, the L1 and L2 regularizers are explained.

4.1.2. L1 or Lasso regularizer

The L1 regularizer (Ng, 2004) penalizes the weight matrix's absolute values from reaching larger values. The most notable advantage of the L1 regularizer is that, for less important features, it decreases the weight value to zero. Therefore, in defining the classifier boundaries, these features are not used. Thus, the L1 regularizer is used to select or reduce features. The mathematical notation for this regularizer is as follows.

$$\lambda \sum_{i=1}^{n} |w_i| \tag{3}$$

λ is regularization parameter. Where n is the number of features in data, and w is the corresponding weight values of each feature.

4.1.3. L2 or Ridge regularizer

Inconstantly, for less important features, the weight value does not need to be zero. The characteristics of corresponding coefficient values are therefore reduced but are kept higher than zero. The square magnitude values are taken from the weight matrix for this purpose, known as the L2 regularizer (Ng, 2004). The mathematical notation of this regularizer is given in Eq. (4). The λ parameter is used to put an extra penalty on the values of corresponding weight values. As it controls the magnitude of the coefficient values, thus, selecting an efficient λ value is a crucial task.

$$\lambda \sum_{i=1}^{n} w_i^2 \tag{4}$$

where n is used for each feature, and w is the coefficient value of each feature.

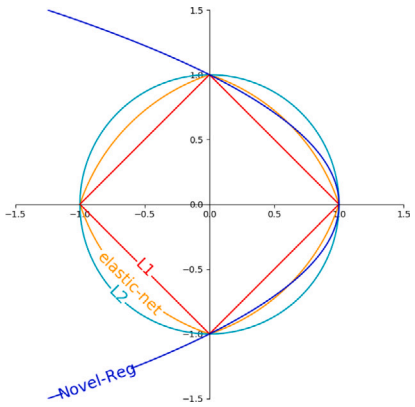


Fig. 2. Contours of L1, L2, elastic-net and new regularizers.

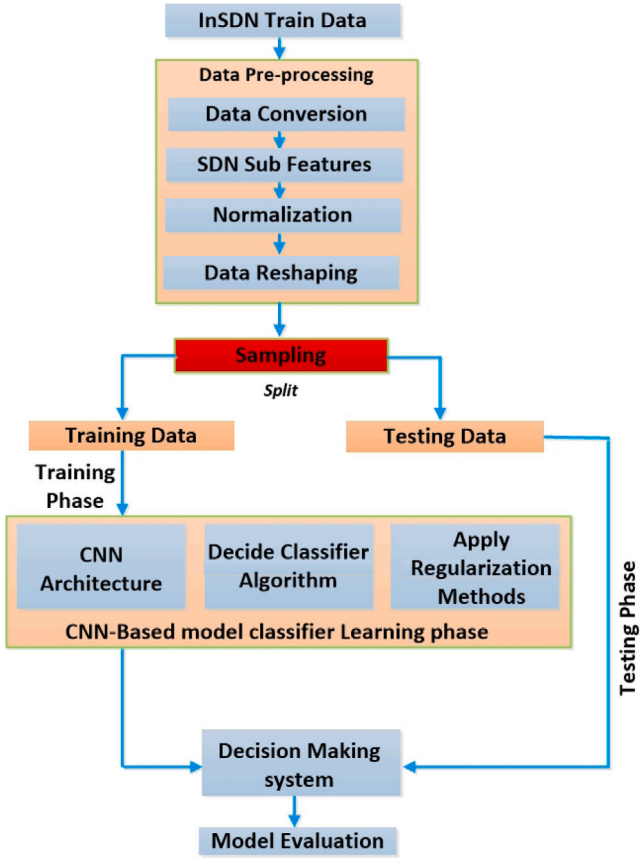


Fig. 3. The diagram flow of the proposed method.

4.1.4. SD-Reg regularizer

L1 regularizer is used for the feature selection or reduction, while L2 gives unimportant features less weight. The major downside of these regularizers is that they control only individual weight values and do not consider the relationship between entries in the weight matrix. We implemented a new regularizer (SD-Reg) to overcome this constraint that considers the weight values' dispersion, known as the standard deviation. To obtain the regularization term, the SD-Reg regularizer takes the standard deviation of the weight matrix and multiplies it by λ. The motive behind this is to create a weight decay adaptive form. Consequently, the regularizer prevents the learning model from taking widespread values from the weight space (See Fig. 2).

In Fig. 2, we have shown the contours of the SD-Reg regularizer. Here, the penalty is equal to 1 for the four all regularizers ($L1$, $L2$, *elastic-net* and the $SD\text{-}Reg$). Further, to maintain the dimensions, we have excluded the sum factor in the proposed regularizer. So, the penalty term λ controls the spread of the regularizer by enforcing constraints. Thus, the learning model does not allow to adapt widespread values from weight space. The mathematical formulation of the new regularizer is given in Eqs. (5)–(7).

$$\lambda\sigma(w) \tag{5}$$

σ denotes the standard deviation of weight values as given below:

$$\sigma(w) = \sqrt{\frac{1}{nk}\left\{\sum_{i=1}^{nk} w_i^2 - \frac{1}{nk}\left(\sum_{i=1}^{nk} w_i\right)^2\right\}} \tag{6}$$

k is the number of rows in the weight matrix and i is the ith row of the weight matrix. The parameter λ is used to control the values of the weight matrix, and n is the number of columns in each ith row of the weight matrix. So, n is the size of the weight vector. In other words, the loss function in our case will become:

$$min_w\{f(X, y : w) + \lambda\sigma(w)\} \tag{7}$$

Thus, we minimize the loss function concerning w by a standard deviation to adopt values within a specific range.

4.2. Our proposed models

We propose hybrid DL-based architectures to explore the potential of DL algorithms for anomaly detection and attack classification tasks within the SDN environment. The proposed models combine the CNN architecture with the ML algorithms (SVM, KNN, and RF). The CNN is employed to extract the deeper representations of the data features, while the classification task is performed using ML algorithms. Fig. 3 summarizes the detailed process of our proposed models. At the first stage, several peprocessing steps (detail illustration about the preprocessing is discussed in Section 5.4) are used to fit the input data for the DL model. CNN, however, is exclusively used for image data. By converting the network data i.e. non-image data, into an image, CNN can utilize its special properties of being computationally efficient and locally focused. We convert the input data from one-dimensional into a two-dimensional matrix. The dimension of the input image is 8×6 or 3×3 for a subset of 48 and 9, respectively. This paper only generates one type of image dataset i.e. a grayscale image, so we set the number of channels into 1. After the input data is prepared, the CNN learning model is initialized to train in input flow. The deep structure of the CNN model used in this study is depicted in Fig. 4.

Tuning the hyper-parameters plays a vital role to determine the performance for the CNN network models. Several parameters including the number of convolutional layers, number of filters, filter size, stride, padding method, and batch size are very important to get the highest performance of the deep CNN model. In practice, there is no magic rule for choosing the optimal values of hyper-parameters, i.e. numerous experiments and trials are the suggested way to find the optimal model structure (Kim et al., 2020). Hence, during the implementation, we tested several kinds of scenarios and combinations to find out the optimal hyper-parameters. Table 1 shows the considered hyper-parameters for our CNN model. The number of convolutional layers relies on the characteristic of the input image. Hence, we used two convolutional layers with output dimensions of 32 and 64, respectively. The filter of 3×3 size is utilized for each layer with a stride equal to 1. Another max-pooling layer of 2×2 size and stride of 1 followed the second convolutional layer. While each convolutional layer learns the feature representation of the previous output, the pooling layer minimizes the dimensionality of the feature map. The output from the pooling layer is reshaped (i.e. flatten layer) for a fully connected layer with a number of neurons equal to 128. The nonlinear mapping function Relue is used for all layers before the output layer. Finally, the classification layer is used to classify the input traffic into normal or attack class. We implement a

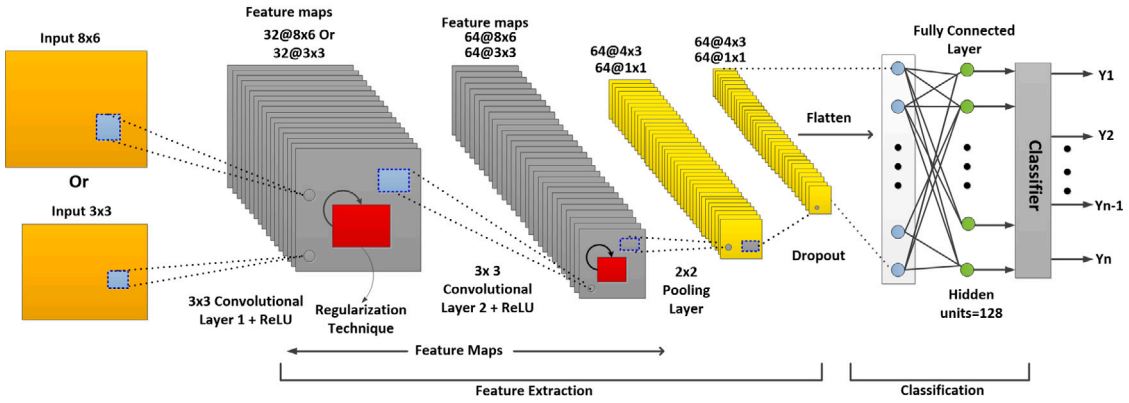


Fig. 4. The structure of CNN model.

Table 1
The optimal hyper-parameters used for our CNN model.

Hyper-parameters	Optimal values
Convolutional layers	2
Number of filters	32, 64
Kernel size	3×3
Stride	1
Pooling layer	Max (2×2)
Padding method	Same
Non-Linear function	Relue
Batch size	128
No of iteration	100
Optimization type	Adam

set of experiments and analysis to compare the performance of different classification methods. In the first experiment, the SoftMax activation function with various regularization methods is used to classify the output features produced from the lower convolutional layers. We also compare the performance of the SD-Reg with L1 and L2 regularizer methods. In the second experiment, we test several hybrid models by replacing the SoftMax layer with three ML techniques, namely SVM, KNN, and RF. While the experimental results demonstrate that the SD-Reg regularizer method provides a high performance than the old L1 and L2 methods, so we used it in all hybrid model experiments. Additionally, to reduce the possibility of overfitting, we use a dropout layer before the fully connected layer. After the model is trained, the testing data is used to evaluate the model and demonstrate how it can work in unseen data.

5. Experiments and evaluation

This section briefly discusses the experimental setup and explains the classification results of our CNN-based models. A set of experiments and analysis were preformed on the InSDN dataset using a series of different approaches: (1) the experimental work was performed on two different set of features that can be extracted through the SDN controller; (2) multiple types of network intrusion classification tests i.e. binary and multi-class classifications are implemented to visualize the performance of the model; (3) the performance of CNNs in unknown attack detection is compared with various ML methods; (4) For more elaboration, various metrics such as the precision, recall, F-score, and receiver operator characteristics curve (ROC) graph are used (cf. Section 5.6).

5.1. Experiment datasets

The performance of the IDS techniques significantly relies on the quality of the training datasets. However, the availability of the benchmark dataset for the intrusion detection domain is one of the main issues that interrupt the development of anomaly detection systems. In different domains such as language translation or biomedical field, we can find massive amount of datasets to evaluate different ML techniques. However, privacy and security issues are the main reasons for the lack of network intrusion detection datasets. The network datasets contain sensitive information, and the public availability of this data can reveal the customer information to the public. Besides, most available datasets are outdated, laboriously anonymized and lack to cover the modern attacks found in the network today. In addition to the aforementioned problem, to our best knowledge, there is no publicly available dataset for testing and evaluation of IDS in SDNs environment. The majority of anomaly detection work in SDNs has implemented standard datasets generated based on the conventional network. However, the characteristics of network traffic in SDN networks is quite different from those of legacy environments. The centralized architecture of the SDN is more vulnerable to attacks that does not exist in legacy networks.

For example, decoupling the SDN controller from the network devices increases the attacker chances to carry out various types of attacks in data communications systems or on the SDN controller itself.

However, such attacks are hard to be detected since the attacker is connected to the victim server in an authorized way.

To tackle all of these problems, we used our InSDN dataset (Elsayed et al., 2020c) to test the performance of the proposed DL models. The InSDN dataset covers recent common attack types such as DoS, DDoS, Probe, Botnet, Exploitation, Password-Guessing, and Web attacks. In addition, the normal traffic in the InSDN dataset covers popular application services such as HTTPS, HTTP, DNS, Email, FTP, SSH. The dataset is available in two different formats i.e. PCAP file and CSV file format, where more than 80 statistical features are extracted using CICFlowMeter open-source tool (Draper-Gil et al., 2016). The source of attacks in the dataset comes from internal and external networks to mimic the real attack scenarios. The dataset has a total of 361,317 instances for normal and attacks traffic, which consists of 68,424 for normal instances, and 292,893 for attack traffic instances. The distribution of these data instances is represented in Table 2.

We can find a disproportionate ratio of observations in each class. The amount of samples for U2R, Web-Attack, and botnet are too few, which will be ignored by most ML techniques, resulting in poor performance. The imbalanced structure is formulated based on the following equation (Karatas et al., 2020):

$$\text{Imbalance Ratio (R)} = \frac{the\ majority\ class\ (max)}{the\ minority\ class\ (min)} \tag{8}$$

We can find that the R value in the original dataset is 4024, which indicates a large gap between the data classes. Consequently, the performance of the intrusion detection model is significantly influenced. The attackers always focus on developing the minority classes to access the target system in an unauthorized way. To address this problem,

Table 2
Proportion of categories in the InSDN dataset.

Label distribution	Samples	%
Normal	68,424	18.93
Probe	98,129	27.15
DDoS	123,942	34.3
DoS-HULK	34,942	9.67
DoS-HTTP-flood	21,038	5.82
DoS-TorshHummer	13,064	3.615
Password-Guessing	1405	0.388
Web-Attack	192	0.053
BOTNET	164	0.0453
U2R	17	0.0047
Total	**361,317**	–

Table 3
The new proportion of categories in the InSDN dataset.

Label distribution	New samples	%
Normal	34212	24.15
Probe	24885	17.56
DDoS	22686	16.01
DoS-HULK	20965	14.80
DoS-HTTP-flood	21038	14.85
DoS-TorshHummer	13064	9.22
Password-Guessing	2052	1.44
Web-Attack	1368	0.965
BOTNET	684	0.482
U2R	684	0.482
Total	**141,638**	–

in the multi-classification training, we combined both oversampling (i.e. SMOTE) and undersampling techniques to duplicating examples in the minority classes and deleting randomly selecting examples from the majority classes, respectively. Applying both techniques can improve the overall performance compared to performing one. We perform random undersampling on normal, probe, DDoS and DoS-HULK flows, while random oversampling is applied on U2R, Web-Attack, and botnet flows. We increase the samples in the minority classes but not the same as the majority classes since the oversampling has the risk to trigger overfitting due to the repetition of the same observations in the minority class. After we applied the oversampling and undersampling, the R value is decreased to 50, which is acceptable to increase the efficiency of the system. The new proportion of categories in the InSDN dataset after applying random oversampling and undersampling is reported in Table 3. In Binary classification, all the benign and attack flow are taken as the original data.

5.2. SDN specific features

This section describes some representative features that can actually be collected in an SDN environment. In SDNs, the SDN controller can collect only statistical features from the data plane devices through API queries. Furthermore, some new features can be manually obtained from the statistics exposed by the controller. In this article, we evaluated the CNN-based models using two different subsets of features.

- In the first experiment, the 48 features are selected from our dataset. These features are chosen based on the framework method of Krishnan et al. (2019), where using too many (useless) features may experience a high computational cost and make the IDSs are not sufficient for a wide range of application. However, the proposed framework of Krishnan et al. (2019) evaluated their learning model with a subset of 50 features since the source and destination IP addresses were selected in their computation. Our approach excluded these two features since the IP address can be changed from one network to another. In addition, the attacker machine can be assigned with the same IP address as the legitimate hosts. Thus, training the learning classifier with such attributes can bias the model toward these features since the IP addresses fail to distinguish between normal and attack traffic. The subset of the 48 features are depicted in Table 4.
- In the second experiment, only 9 features have been used to evaluate the approach models. The main objective of implementing a few features for DL-based classifiers is to minimize the cost of higher processing power and memory requirements and still achieve accuracy and speed. Table 5 shows the subset of 9 features that implemented for the further evaluation.

Table 4
The subset of 48 features.

No.	Attribute name	No.	Attribute name
1	Protocol	25	Fwd-IAT-Min
2	Flow-duration	26	Bwd-IAT-Tot
3	Tot-Fwd-Pkts	27	Bwd-IAT-Mean
4	Tot-Bwd-Pkts	28	Bwd-IAT-Std
5	TotLen-Fwd-Pkts	29	Bwd-IAT-Max
6	TotLen-Bwd-Pkts	30	Bwd-IAT-Min
7	Fwd-Pkt-Len-Max	31	Fwd-Header-Len
8	Fwd-Pkt-Len-Min	32	Bwd-Header-Len
9	Fwd-Pkt-Len-Mean	33	Fwd-Pkts/s
10	Fwd-Pkt-Len-Std	34	Bwd-Pkts/s
11	Bwd-Pkt-Len-Max	35	Pkt-Len-Min
12	Bwd-Pkt-Len-Min	36	Pkt-Len-Max
13	Bwd-Pkt-Len-Mean	37	Pkt-Len-Mean
14	Bwd-Pkt-Len-Std	38	Pkt-Len-Std
15	Flow-Byts/s	39	Pkt-Len-Var
16	Flow-Pkts/s	40	Pkt-Size-Avg
17	Flow-IAT-Mean	41	Active-Mean
18	Flow-IAT-Std	41	Active-Std
19	Flow-IAT-Max	43	Active-Max
20	Flow-IAT-Min	44	Active-Min
21	Fwd-IAT-Tot	45	Idle-Mean
22	Fwd-IAT-Mean	46	Idle-Std
23	Fwd-IAT-Std	47	Idle-Max
24	Fwd-IAT-Max	48	Idle-Min

Table 5
The subset of 9 features.

No.	Attribute name
1	Protocol
2	Flow Duration
3	Tot Fwd Pkts
4	TotLen Fwd Pkts
5	Fwd Pkt Len Mean
6	Flow Byts/s
7	Flow Pkts/s
8	Flow IAT Std
9	Fwd IAT Mean

5.3. Data visualization

In order to understand the distribution of data intuitively, the effective t-SNE algorithm (Maaten and Hinton, 2008) is used to better visualize and explore the high-dimensional data. In our experiment, 10000 connection records are randomly chosen from the InSDN dataset and passed through Principal Component Analysis (PCA) to reduce its dimensional features. Using fewer features will reduce the relation between them and facilities the visualization of the data . The distribution of the data in low space is represented in Fig. 5. From the graph, it can be clear that the cluster of the attacks is not spatially separated from each other. However, the classical ML algorithms do not inclusively work better with this kind of data.

5.4. Data preprocessing

In order to build an accurate IDS model, a few steps should be taken before fed the data into the learning classifiers. The pre-processing

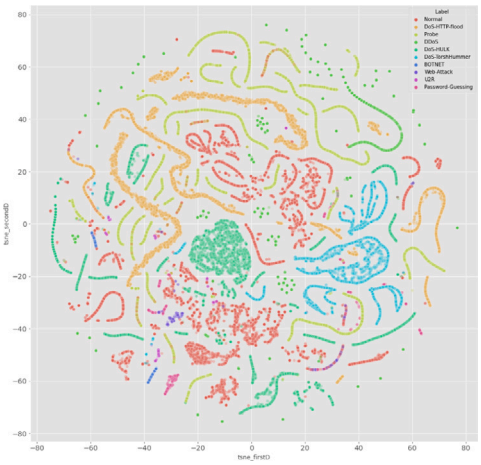


Fig. 5. Visualization result of InSDN data flow.

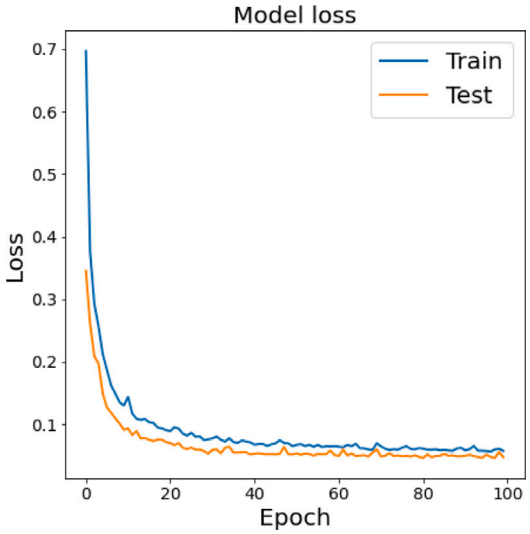


Fig. 6. Trend of training and validation loss.

is essential to create an efficient IDS approach and to reduce the computationally intensive operations. In this work, we perform some tasks to prepare the data as follows:

- The features have different ranges, so we rescaling the data using the standardization method (Z-score normalization) based on Eq. (9) to transform the data into a different scale, where the standardized data have a mean value equal 0 and standard deviation of 1.

$$x(i) = \frac{x(i) - mean(x(i))}{standard\_deviation(x(i))} \tag{9}$$

- For all experiments, the dataset is divided into 70% for training and 30% for testing purpose using Train_test_split method from Scikit-Learn library with test_size = 0.3.
- The labeled string is encoded with a numerical value to transform the label class into a unique integer representation using the one-hot encoding technique. This article includes binary and multiclass classification. Therefore, in binary experimental, the normal string was assigned to a binary value of 0, and all malicious traffic has a value of 1, while in multiclass, each attack is mapped to a unique digit value.
- Data Reshaping. The input of a CNN should be in the image format with three-dimension i.e. height, width, channel. However, the network traffic is in a one-dimensional data, which is not compatible with the CNN. An extra step is required to change the format of the input traffic to fit the resolution dimensions of CNN. In the case of the subset 48 features, we convert the 48-dimensional vector into images with 8 × 6 shape, while an image with 3 × 3 was created for 9-dimensional vector input. In this paper, only a grayscale image which has a single channel is generated, so we set the channel number into 1.

5.5. Experimental setup

The experiment was designed and executed using Python programming language, where Keras with Tensorflow backend library is used for all proposed approaches. Table 6 describes the experimental configuration used to evaluate the model parameters.

5.6. The evaluation metrics

We used the most common performance measures like the accuracy, precision, recall, and F-score metrics to evaluate the performance of all models, as the following equations.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

where, TP (true positive) and TN (true negative) represent the correctly predicted values, while FP (false positive) and FN (false negative) indicate misclassified events.

6. Experimental results and analysis

This section provides a detailed analysis of the results obtained using our proposed models. The performance of the proposed models is evaluated on the InSDN dataset by conducting a series of different experiments.

6.1. The results of 48 sub-SET features

In this section, we evaluate our proposed models using a subset of 48 features that can actually be collected in an SDN environment.

6.1.1. Loss trend of our proposed model

The trend of training and testing loss over the number of epochs for CNN-based new regularization model is depicted in Fig. 6. It can be found that the loss trends for training and validation sets tight to each other and converge after a few hundreds of epochs. We also noticed that the testing loss is lower than the training loss; this is because the dropout layers is used for the model training. The dropout layer introduces noise that is not injected during the test period. As a result, the error will be less in case of testing due to the better generalization to combat the overfitting problem.

Table 6
Experimental environment.

Operating System	Windows 10 pro 64-bit
Memory	16 GB
CPU	Intel(R) UHD Graphics 620, I7-8650U CPU @ 1.90 GHz (8 cores), 2.1GHz
Anaconda	4.9.2
Python	3.7.0
Keras	2.4.2
Tensorflow	2.2.0

Table 7
Precision, recall, and F1-score of the different CNN methods in case of the subset of 48 features.

Technique	Evaluation results %	Binary-class		Multiclass									
		Normal	Attack	Normal	BOTNET	DDoS	DoS-HTTP-flood	DoS-HULK	DoS-TorshHummer	Password-Guessing	Probe	U2R	Web-Attack
CNN-Softemax (L1)	Precision	99.04	97.38	99.41	97.58	99.92	99.52	98.26	98.99	96.68	93.98	99.45	74.33
	Recall	94.69	99.81	94.71	100	99.98	99.96	99.67	100	69.80	99.53	100	83.58
	F1-score	97.08	98.58	97.00	98.77	99.95	99.74	98.96	99.49	81.07	96.67	99.72	78.68
CNN-Softemax (L2)	Precision	99.43	97.65	99.43	99.01	99.92	99.57	99.67	99.71	96.09	93.81	98.91	80.66
	Recall	95.25	99.72	96.01	100	99.98	99.98	99.51	99.56	74.76	99.54	100	84.07
	F1-score	97.30	98.68	97.69	99.50	99.95	99.77	99.59	99.64	84.09	96.59	99.45	82.33
CNN-Softemax (SD-Reg)	Precision	99.80	98.27	99.93	99.01	100	99.84	99.80	99.79	97.79	94.30	98.91	83.95
	Recall	96.51	98.27	96.21	100	99.98	99.98	99.87	100	77.79	99.76	100	95.02
	F1-score	96.51	98.27	98.04	99.50	99.99	99.91	99.83	99.89	86.65	96.95	99.45	89.14
CNN-SVM	Precision	99.69	98.50	99.41	100	100	99.34	100	100	77.77	95.85	100	78.57
	Recall	97.33	99.84	96.44	100	100	100	100	100	70.00	98.93	100	91.66
	F1-score	98.50	99.24	98.03	100	100	99.67	100	100	73.68	97.36	100	84.61
CNN-KNeighbor	Precision	99.09	98.50	98.69	98.53	99.97	99.82	99.80	99.69	94.40	96.90	100	88.55
	Recall	97.04	99.54	97.48	100	99.98	100	99.85	99.92	92.97	97.92	100	94.27
	F1-score	98.05	99.02	98.08	99.26	99.97	99.91	99.83	99.80	93.19	97.41	100	91.32
CNN-RF	Precision	98.74	99.56	99.45	99.01	100	99.84	99.80	99.77	98.23	96.29	100	84.33
	Recall	99.15	99.35	97.09	100	99.98	100	99.87	100	88.65	98.61	100	97.76
	F1-score	98.95	99.45	98.19	99.50	99.99	99.92	99.83	99.88	94.17	97.44	100	90.55

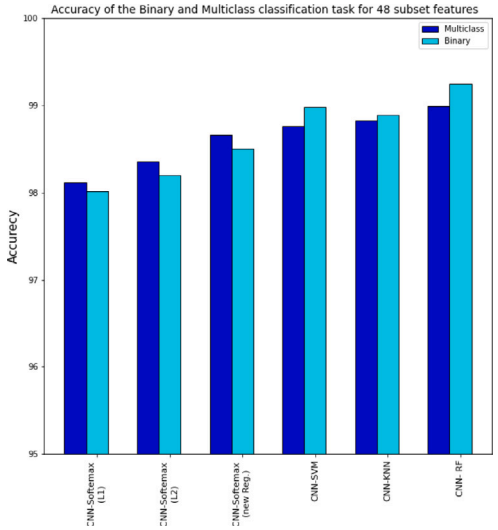


Fig. 7. Accuracy of the binary and multiclass classification task for 48 subset features.

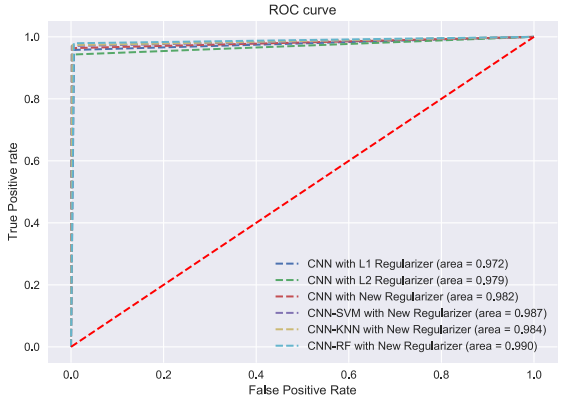


Fig. 8. Receiver Operating Curve (ROC) of binary CNNs approaches using 48 sub Features.

6.1.2. Performance comparisons

Fig. 7 shows the graphical view of detection accuracy for a subset of 48 features for proposed CNN models on InSDN dataset, while the comparison of precision, recall and F1-measure of different classifiers are represented in Table 7. The estimated results indicate that the hybrid models, CNN-SVM, CNN-KNN and CNN-RF give the higher results in all measurements than the single CNN models for binary and multiclass calcification. The combination of CNN and RF algorithm provides the best accuracy up to 99.28%, 98.92%, for binary and multiclass, respectively. Further, the CNN-RF model gains the highest percentage in precision, recall and F-score among other approaches for most attack classes. We can also note that the CNN-based on SD-Reg method imparted an excellent role in detection accuracy and other measurements compared to L1, L2 based regularization methods. The accuracy of the multiclass for single CNN models is slightly higher than the binary classification. On the other hand, we can find the hybrid models have higher accuracy in binary classification than the multiclass. In addition, the majority of attack classes have very high score metrics for all learner classifiers, except the results of web-attack and password-guessing, which give poor performance metrics. This is due to the fact that the high degree of similarity of those attacks with the normal traffic since the attacker access the target system in an authorized manner. Thus, the learning classifier can impose complexity in defending the desired decision boundaries.

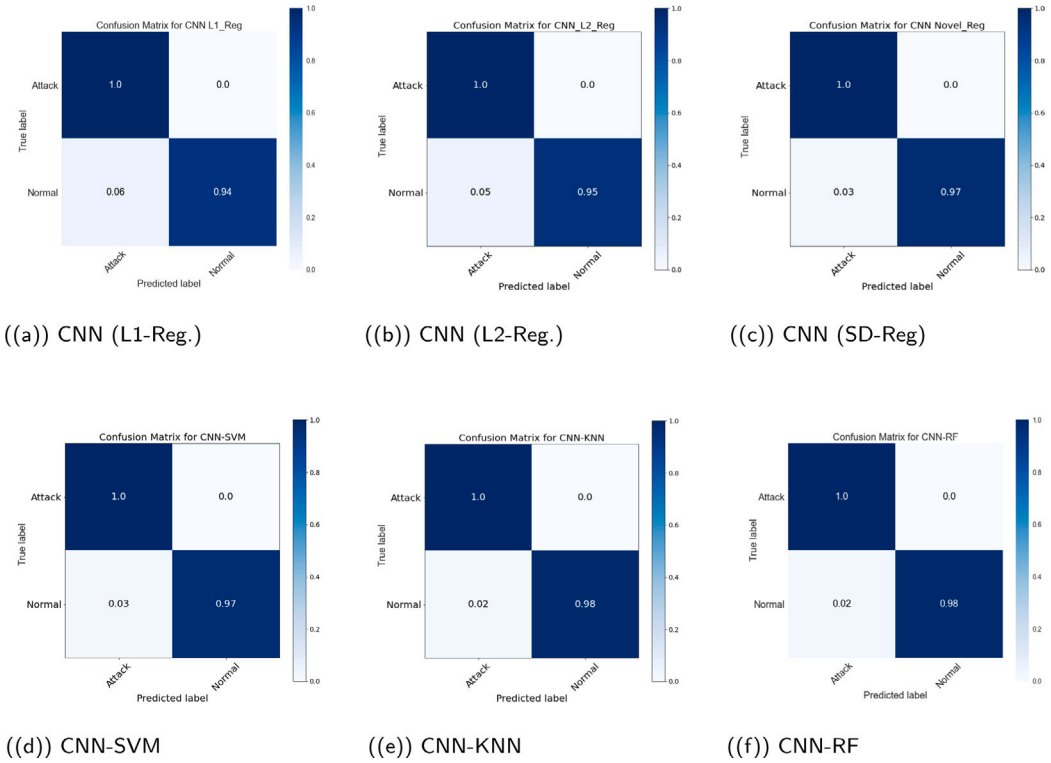


Fig. 9. Confusion matrix for binary classification obtained from CNNs approaches for subset of 48 features. All hyper models are trained using the new regularization technique.

6.1.3. Receiver operating characteristic (ROC)

To further measure the performance of DL approaches, the Receiver operating characteristics (ROC) curve is used, as depicted in Fig. 8. It shows in a graphical way the trade-off between true positive and false positive rates. In addition, the area under the ROC curve gives how the model performs in general, where a greater area means the model is significantly able to distinguish 0s as 0s and 1s as 1s. In contrast, The AUC near to the 0 indicated the model has the worst measure of separability. The hybrid CNN-RF gives a value of 0.99, which indicates that 99% of positive and negative rates are successfully separated.

6.1.4. Confusion matrix

The confusion matrix is conducted for the further evaluation of models performance. It summarizes the correct and false predictions. The binary and multiclass confusion-matrices for all models are represented in Figs. 9 and 10, respectively.

6.2. The results of 9 sub-SET features

Similar to the previous experiments, we test the same CNN approaches using a subset of 9 features instead of 48 features. The main objective behind this evaluation is to represent how the CNN models can perform better with less number of features. The proposed detection model should be lightweight to reduce computational cost while having relatively high accuracy. As a result, it can be applied directly to the SDN without consuming the network resources or cause any delay for legitimate users to access their target system. The detailed results for binary and multiclass classification for a subset of 9 features are represented in Figs. 11, 12, 13, 14 and Table 8. Similar to the previous results of 48 sub-features, CNN based on SD-Reg outperforms the old regularization methods (i.e. L1 and L2). The overall score metrics for SD-Reg are very high for the majority of attacks. Furthermore, the AUC value of the SD-Reg method is 0.944, which is higher than L1 and L2 methods.

In addition, hybrid models achieved better results than single CNN models. It can be found that the CNN-RF provides the highest performance for binary and multiclass classification in all evaluation metrics. The average accuracy of binary and multiclass classification for the CNN-RF model is 97.47% and 97.37%, respectively. This demonstrates the strength and the potential of CNN for efficient anomaly detection using a minimum amount of features. Another observation is that the single models have good results in the binary classification over the multiclass classification using 9 features for training. This is because few features have been used for training and are not enough to distinguish between different attack classes. Also, we can see that the CNN-RF retains the highest AUC score with a value of 0.968, followed by CNN-KNN and CNN-SVM models, with values of 0.967 and 0.951, respectively.

6.3. Detection of unknown attacks

In this section, the potential of the proposed models was demonstrated to detect unknown attacks without a pr-defined knowledge. The dataset contains 9 different types of attacks. We conduct several experiments where the hybrid learning models will be tested each time with a new attack type, while the remainder attacks are used during the training phase. In each experiment scenario, the performance of hybrid DL models has been compared with several classical ML algorithms. The algorithms used in this evaluation include Logistic Regression (LR), NB, SVM, KNN, DT, and RF. The default setting from Skit-learn is used for classical ML methods. In addition, we compare our models with the LSTM algorithm to further represent the dominance of our model in unseen attack detection. The LSTM is a type of recurrent neural network (RNN) algorithm, crucial for anomaly detection, that outperforms the traditional neural network models, as proved in several research works Luo et al. (2017), Malhotra et al. (2015). The performance of different classifiers are depicted in Figs. 15, 16 and Table 9. We can see from the reported results (Table 9), that the LSTM outperformed most of the traditional machine learning algorithms. However, it is clear from these results that the overall performance of our proposed deep

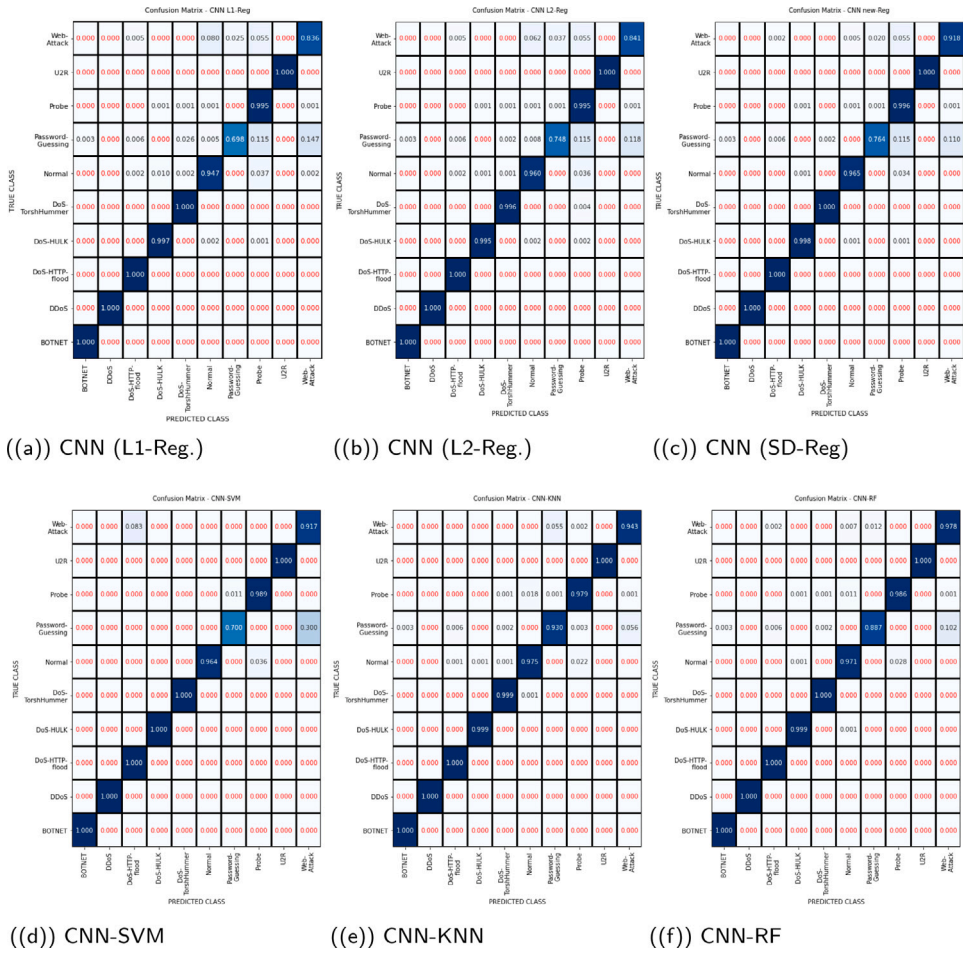


Fig. 10. Confusion matrix for multiclass classification obtained from CNNs approaches for subset of 48 features.

Table 8
Precision, recall, and F1-score of the different CNN methods in case of the subset of 9 features.

Technique	Evaluation results %	Binary-class		Multiclass									
		Normal	Attack	Normal	BOTNET	DDoS	DoS-HTTP-flood	DoS-HULK	DoS-TorshHummer	Password-Guessing	Probe	U2R	Web-Attack
CNN-Softemax (L1)	Precision	99.05	93.13	98.12	92.59	99..48	96.06	88.49	84.28	94.94	79.22	85.71	69.58
	Recall	85.43	99.59	85.22	49.50	99.82	99.87	80.78	93.22	42.01	99.33	42.62	41.54
	F1-score	85.43	96.25	91.21	64.51	99.65	97.93	84.46	88.52	58.25	88.14	56.93	52.02
CNN-Softemax (L2)	Precision	98.72	94.03	98.52	70.83	99.76	97.86	90.65	83.26	90.17	84.73	79.47	87.09
	Recall	87.49	99.43	87.57	50.49	99.80	99.58	84.48	96.51	64.53	99.14	82.51	40.29
	F1-score	92.77	96.65	92.72	58.95	99.78	98.71	87.45	89.39	75.23	91.37	80.96	55.10
CNN-Softemax (SD-Reg)	Precision	99.59	94.72	98.18	81.89	99.82	98.80	91.60	95.90	94.79	82.66	91.34	94.70
	Recall	88.97	99.81	90.41	100	99.94	99.93	91.60	94.55	68.03	99.50	94.61	46.88
	F1-score	93.98	97.20	94.13	90.04	99.88	99.36	87.91	94.55	79.21	90.30	92.95	62.71
CNN-SVM	Precision	97.87	95.73	97.58	85.95	99.91	99.55	90.12	98.79	98.39	91.74	94.65	98.43
	Recall	91.25	98.99	92.38	100	99.97	99.69	95.86	96.10	68.37	99.17	96.72	47.01
	F1-score	94.44	97.33	94.91	92.44	99.94	99.62	92.90	97.43	80.67	95.31	95.67	63.63
CNN-KNeighbor	Precision	97.87	97.20	96.16	98.04	99.95	99.45	95.89	99.07	87.82	94.61	95.72	78.91
	Recall	94.35	98.96	94.47	99.50	99.97	99.64	97.53	98.54	76.03	97.01	97.81	72.63
	F1-score	96.08	98.07	95.31	98.77	99.96	99.55	96.70	98.81	81.50	95.80	96.75	75.64
CNN-RF	Precision	97.63	97.39	97.15	97.11	99.94	99.49	96.51	99.28	94.14	94.63	96.25	80.83
	Recall	94.75	98.84	94.55	100	99.97	99.69	98.01	98.80	76.99	98.50	98.36	76.61
	F1-score	96.17	98.11	95.84	98.53	99.95	99.59	97.25	99.04	84.71	96.53	97.29	78.67

hybrid models is higher than the classical ML and the LSTM techniques for all the types of attacks in the two subsets of features. In addition, CNN-SVM has higher performance than CNN-KNN in the case of 48 dataset version evaluation. When less features are used i.e. for the 9 subset version, the CNN-SVM performance becomes low. However, the CNN-RF is still the robustness approach for the majority of attacks.

Regrading the single ML techniques, the LR and NB algorithms have the worst accuracy for all attacks, followed by SVM. In contrast, the LSTM, RF and DT have the better accuracy, followed by KNN algorithm.

Table 9
Precision, recall, and F1-score of different techniques for unknown attacks detection.

	Technique	Evaluation results (%)	BOTNET	DDoS	DoS-HTTP-flood	DoS-HULK	DoS-TorshHummer	Password-Guessing	Probe	U2R	Web-Attack
48-sub feature	LR	Precision	91.19	89.80	91.96	91.94	94.38	92.32	90.96	93.35	88.96
		Recall	99.85	99.70	99.57	100	100	99.84	100	99.72	100
		F1-score	95.32	94.49	95.62	95.80	97.10	95.94	95.26	96.43	94.16
	NB	Precision	90.39	89.30	91.72	92.65	92.77	92.57	89.75	93.44	89.39
		Recall	97.77	96.95	96.35	99.09	97.51	97.73	97.98	97.21	98.18
		F1-score	93.94	92.97	93.98	95.76	95.08	95.08	93.68	95.29	93.58
	SVM	Precision	91.57	91.22	93.28	92.07	94.24	93.76	91.69	94.19	91.65
		Recall	100	99.56	99.43	100	99.44	99.84	99.37	99.30	99.69
		F1-score	95.60	95.21	96.26	95.87	96.77	96.71	95.38	96.68	95.50
	KNN	Precision	96.78	96.60	96.98	97.76	98.21	97.19	97.72	97.66	96.17
		Recall	98.36	99.12	99.15	99.24	99.03	99.24	99.84	98.74	98.94
		F1-score	97.57	97.85	98.05	98.50	98.62	98.20	98.77	98.20	97.53
	DT	Precision	98.50	97.98	98.04	98.50	99.30	99.08	99.22	98.34	97.88
		Recall	97.92	98.69	98.59	99.24	98.89	98.34	99.53	99.44	98.18
		F1-score	98.21	98.33	98.32	98.87	99.10	98.71	99.37	98.89	98.03
	RF	Precision	98.51	98.55	98.19	98.89	99.28	99.04	99.84	98.61	98.04
		Recall	98.66	98.69	99.43	99.01	99.01	98.89	99.84	99.40	98.48
		F1-score	98.59	98.62	98.81	98.94	99.14	98.96	99.84	99.01	98.26
	LSTM	Precision	98.21	97.91	98.32	98.23	98.54	98.31	99.85	98.13	98.23
		Recall	99.97	99.98	99.87	99.62	99.92	99.87	99.88	99.86	99.46
		F1-score	99.08	98.93	99.09	98.92	99.23	99.08	99.87	98.99	98.84
	CNN-SVM	Precision	98.20	98.05	98.52	98.36	98.96	98.74	99.88	98.25	98.10
		Recall	99.95	99.91	99.91	99.92	99.73	99.61	99.94	99.97	99.96
		F1-score	99.07	98.97	99.21	99.13	99.34	99.18	99.91	99.10	99.02
	CNN-KNeighbor	Precision	98.63	98.23	98.60	98.36	98.95	98.55	99.95	98.58	98.58
		Recall	99.62	99.77	99.53	99.78	99.72	99.71	99.93	99.69	99.46
		F1-score	99.13	98.99	99.07	99.07	99.33	99.13	99.94	99.13	99.02
	CNN-RF	Precision	98.44	98.10	98.47	98.29	99.21	98.70	99.98	98.80	98.53
		Recall	99.75	100	100	99.85	99.12	99.71	100	99.87	99.46
		F1-score	99.09	99.04	99.23	99.07	99.17	99.20	99.99	99.34	98.99
9-sub feature	LR	Precision	83.35	89.23	87.80	86.64	87.74	88.01	84.27	87.17	83.35
		Recall	98.76	99.71	98.98	99.84	99.58	100	99.53	99.40	98.76
		F1-score	90.40	94.18	93.06	92.77	93.29	93.62	91.27	92.88	90.40
	NB	Precision	77.61	90.65	76.22	74.71	78.97	77.06	85.83	79.63	77.61
		Recall	97.22	98.40	96.52	99.08	97.24	98.38	93.80	96.86	97.22
		F1-score	86.32	94.37	85.18	85.19	87.16	86.43	89.64	87.40	86.32
	SVM	Precision	83.83	89.90	90.10	86.54	88.37	89.37	84.94	88.14	83.83
		Recall	98.30	99.42	98.84	100	99.44	99.70	99.53	98.80	98.30
		F1-score	90.49	94.42	94.27	92.78	93.58	94.25	91.66	93.17	90.49
	KNN	Precision	94.73	95.90	95.37	94.75	94.89	94.78	94.31	93.86	94.73
		Recall	97.07	98.40	98.55	99.08	97.38	98.53	97.52	98.20	97.07
		F1-score	95.89	97.13	96.93	96.87	96.12	96.62	95.89	95.98	95.89
	DT	Precision	96.90	96.25	97.25	96.68	96.69	95.57	94.11	96.14	96.73
		Recall	96.45	96.81	97.25	97.71	96.69	97.95	99.92	96.71	95.99
		F1-score	96.67	96.53	97.25	97.19	96.69	96.74	96.93	96.42	96.36
	RF	Precision	95.44	97.39	95.13	97.57	97.90	96.01	97.53	96.18	96.39
		Recall	99.60	97.68	99.71	98.01	96.41	98.68	97.98	97.91	97.42
		F1-score	97.47	97.53	97.36	97.79	97.15	97.32	97.76	97.04	96.90
	LSTM	Precision	95.54	96.30	95.08	94.79	95.12	94.42	93.84	94.02	94.02
		Recall	98.85	99.34	99.71	99.46	99.77	99.76	99.77	99.76	99.83
		F1-score	97.17	97.80	97.34	97.07	97.39	97.02	96.72	96.81	96.84
	CNN-SVM	Precision	96.63	96.29	97.55	96.25	96.32	96.03	98.14	95.28	95.93
		Recall	98.76	99.52	97.97	99.60	99.72	99.26	98.45	99.61	97.87
		F1-score	97.68	97.88	97.76	97.90	97.99	97.62	98.29	97.40	96.89
	CNN-KNeighbor	Precision	95.97	97.73	96.41	96.95	97.39	95.95	98.90	95.47	95.37
		Recall	99.15	98.54	99.61	99.50	98.74	99.56	99.07	99.49	99.19
		F1-score	97.54	98.14	97.98	98.21	98.06	97.72	98.99	97.44	97.24
	CNN-RF	Precision	98.29	97.75	96.50	97.70	97.05	96.26	99.87	96.04	98.28
		Recall	97.53	99.17	99.61	99.50	99.30	99.31	99.88	99.14	97.22
		F1-score	97.91	98.45	98.03	98.59	98.16	97.76	99.88	97.56	97.75

6.4. Comparison with different datasets

To significantly test and evaluate the efficiency of the models, comparative work is done using two previously datasets, namely, the CSE-CIC-IDS2018 (Sharafaldin et al., 2018) and the UNSW-NB15 (Moustafa and Slay, 2015). The preprocessing steps for the CSE-CIC-IDS2018 and UNSW-NB15 were illustrated in Karatas et al. (2020) and Pantiukhin et al. (2019), respectively. We investigate the performance of the binary classification for both datasets. Our obtained results for the different models are represented in Table 10. It is clearly shown

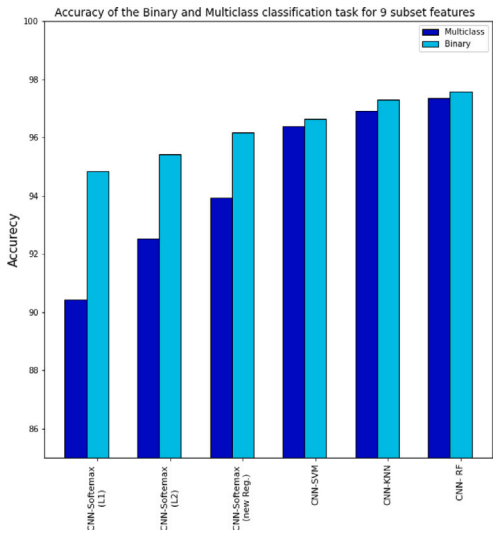


Fig. 11. Accuracy of the binary and multiclass classification task for 9 subset features.

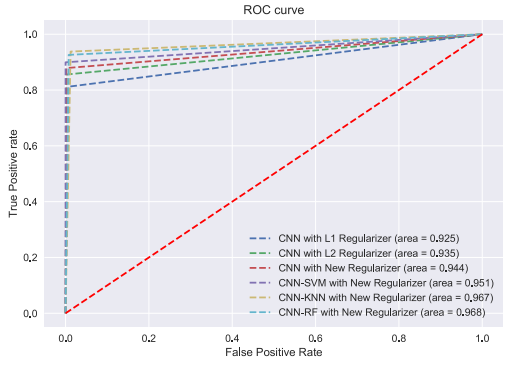


Fig. 12. Receiver Operating Curve (ROC) of binary CNNs approaches using 9 sub features.

that our proposed hybrid models outperformed the single models for both datasets, except for the CNN-SVM, where its performance is similar with the CNN-Softemax using SD-Reg for CSE-CIC-IDS2018. The CNN-KNN model provided the best results for UNSW-NB15, while its performance is similar to CNN-RF in the CSE-CIC-IDS2018. Again, the new Reg. obtains the best results when compared to L1 and L2 regularization, while the L1 provides the worst efficiency for all datasets.

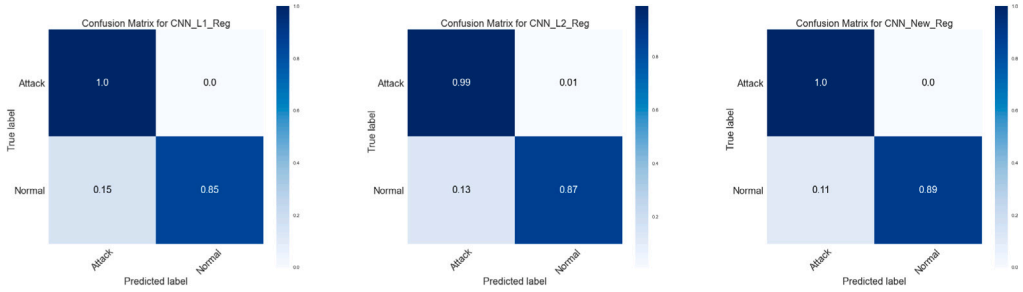
Additionally, we calculated the training and testing time for all models using both datasets. In general, the consuming time relies on several factors such as the dataset size, the number of hidden layers, the number of neurons per each layer, the type of the model, the experiential environment, etc. However, the CNN would be the proper approach to reduce the time consuming since the CNN can effectively address the parameter explosion problem, by allowing the parameters sharing through each layer. Further, although training a deep learning model is a challenging and expensive task, this problem has typically been tackled by using a GPU accelerator. The GPU computational capabilities have grown by over 10X over the past few years (Cui, 2020) — a trend that is expected to continue unabated in the next years with advances in GPU architectures and the use of special-purpose training chips. This trend further reduces the computational times.

In Table 10, we can see that the approaches based on SD-Reg have less consuming time for both training and testing compared to traditional ones i.e. L1 and L2. Moreover, the hybrid models take a long time for the training process, but the time significantly becomes less during the testing. The CNN-RF accounted a higher training time but it has less testing time compared to CNN-SVM and CNN-KNN approaches. However, the long training time does not significantly impact the model functionality. As mentioned in the previous section, figuring out the optimal set of hyper-parameters can be one of the most time consuming portions of creating a machine learning model, and that is particularly true for deep learning. Once we find the right values that work well on the training data and have high performance on the test data, we no longer need to manually schedule the hyper-parameters due to real-time detection. On the other hand, the small detection time indicates the robustness of the hybrid techniques for anomaly detection, especially in virtual environments like SDNs. Such environments are susceptible to delay during their specific structure. Thus, high-speed detection techniques should be implemented to deal with the attacks very fast, before they target the controller and cause crucial consequences on the whole network.

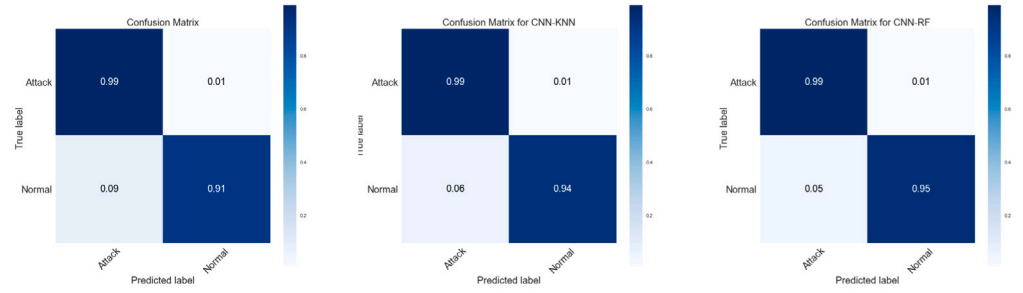
7. Discussion and limitation

This article addresses the security issues in the SDN networks by proposing hybrid DL models based on the CNN algorithm. Whereas, one of the crucial concerns of training ML/DL algorithms is the risk of overfitting. This problem mostly happened when the model tries to learn the noise in the data that does not really represent the true properties of the regular patterns. Although several regularization methods are available to fine-tune this problem and to choose the appropriate model for a better prediction, these methods consider individual values of the weight without emphasizing the relationship between weight matrix entries. As a result, the model can use widespread values from weight space, and therefore, any small change in a feature can cause a large change in the prediction. To address these limitations, we used the SD-Reg method to enhance the performance of DL for unknown attack detection. The experimental results showed that the SD-Reg outperformed the old methods and achieved higher accuracy for binary and multiclass classification. The implementation of our deep learning model included two stages. In the first stage, the feature representation of the input data is obtained from the CNN layers, while the classification tasks are performed in the second stage. The CNN has been used to decrease the number of training parameters, and therefore, we can generate a new model capable of detecting network intrusions with a low computational cost. Besides, the CNN can reduce the dimensional of the input features by using the pooling layer. Several ML algorithms are used for classification scenarios to ensure the accuracy of the DL results. We have designed our CNN model to perform the binary and multiclass classifications to distinguish between 10 categories in the InSDN dataset. Likewise, our model has also several limitations, as follows:

- Although accuracy and evaluations metrics are crucial to evaluate the performance of the model; these metrics are not enough without the real implementation of the developed model under SDN networks. Evaluating the network performance by taking into account throughput, resource utilization, and time delay requirements is very important to intensively test the potential of the proposed intrusion detection. In the near future, we plan to implement the proposed model in real SDN networks and test how the intrusion detection can deal with the detected attacks in real-time.
- While the use of L1 and L2 regularizations is point-less when the dimensional data is very high and the number of training samples is relativity low, our SD-Reg also has some limitations. It is not ideal for feature selection or feature reduction. In a sense, the dataset's complexity increases linearly in the number of irrelevant features by using L2 regularization in logistic regression, while it increases logarithmically by using L1 regularization (Ng, 2004). In a similar learning environment, the main problem with the new regularization penalty is that, as with any dimensional reduction

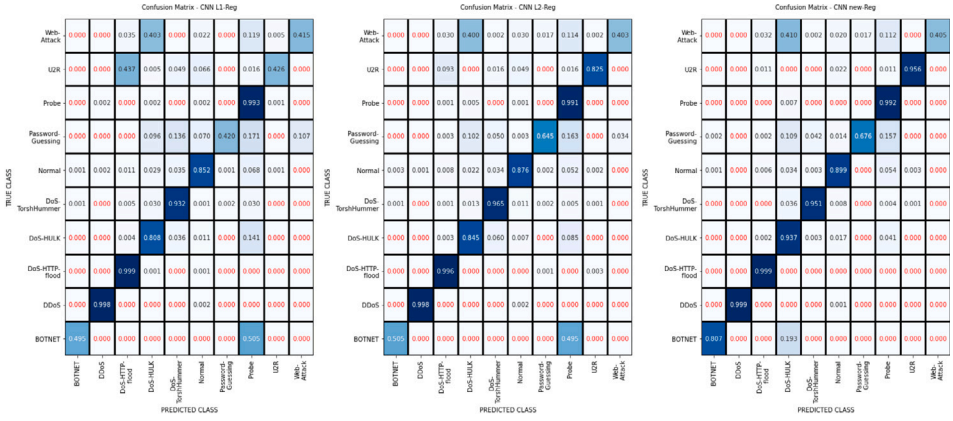


((a)) CNN (L1-Reg.) ((b)) CNN (L2-Reg.) ((c)) CNN (SD-Reg)

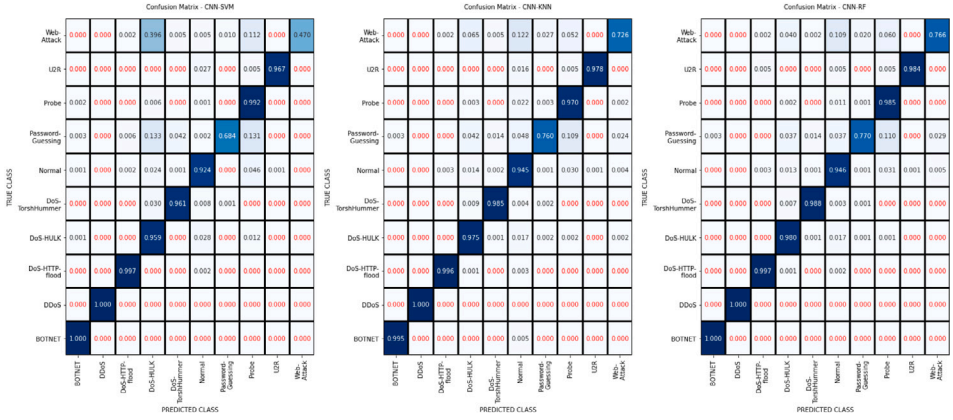


((d)) CNN-SVM ((e)) CNN-KNN ((f)) CNN-RF

Fig. 13. Confusion matrix for binary classification obtained from CNNs approaches for subset of 9 features.



((a)) CNN (L1-Reg.) ((b)) CNN (L2-Reg.) ((c)) CNN (SD-Reg)



((d)) CNN-SVM ((e)) CNN-KNN ((f)) CNN-RF

Fig. 14. Confusion matrix for multiclass classification obtained from CNNs approaches for subset of 9 features.

Table 10
Experimental results on public datasets.

Dataset	Approach	Precision (%)		Recall (%)		F-score (%)		Average accuracy (%)	Training time (s)	Testing time (s)
		Attack	Normal	Attack	Normal	Attack	Normal			
UNSW-NB15	CNN-Softemax (L1)	98.73	99.49	99.43	98.86	99.08	99.17	99.13	3463.094	21.734
	CNN-Softemax (L2)	98.46	99.99	99.99	98.51	99.22	99.24	99.23	3527.359	21.25
	CNN-Softemax (SD-Reg)	98.57	99.96	99.96	98.62	99.26	99.29	99.27	2622.453	20.406
	CNN-SVM	98.60	100	100	98.73	99.29	99.36	99.33	2626.438	0.109
	CNN-KNeighbor	98.98	100	100	99.01	99.49	99.50	99.50	2641.0.32	0.047
	CNN-RF	99.18	99.60	99.59	99.21	99.38	99.41	99.40	2682.719	0.031
CSE-CIC-IDS2018	CNN-Softemax (L1)	99.36	99.28	99.28	99.36	99.32	99.32	99.32	567.359	3.281
	CNN-Softemax (L2)	99.82	99.20	99.19	99.82	99.50	99.51	99.51	578.516	4.766
	CNN-Softemax (SD-Reg)	99.88	99.32	99.31	99.88	99.59	99.60	99.60	454.297	3.312
	CNN-SVM	100	99.20	99.20	100	99.59	99.60	99.60	457.141	0.094
	CNN-KNeighbor	99.59	100	100	99.60	99.79	99.80	99.80	471.438	0.062
	CNN-RF	100	99.60	99.60	100	99.79	99.80	99.80	514.391	0.031

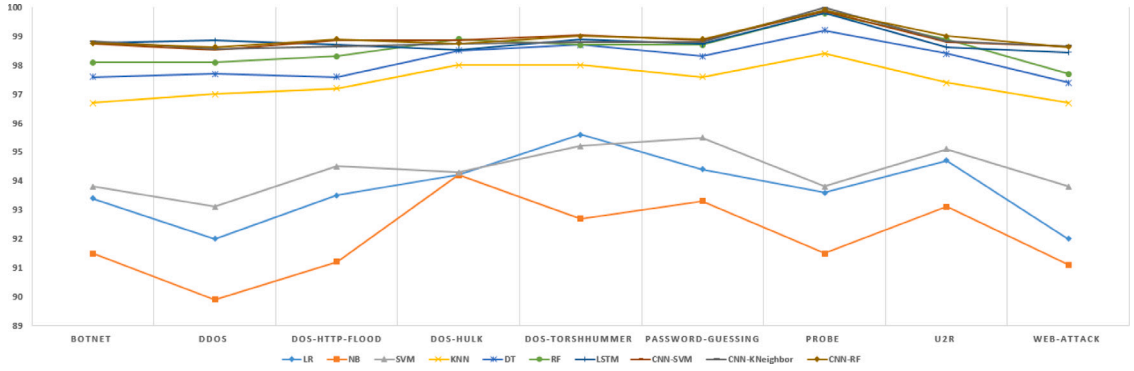


Fig. 15. The Average Accuracy of different techniques of 48-subset features.

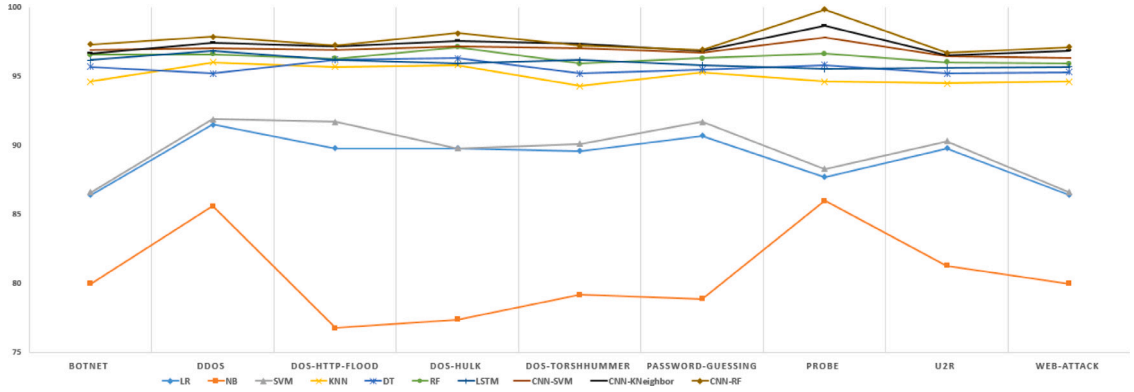


Fig. 16. The Average Accuracy of different techniques of 9-subset features.

algorithm, certain important independent variables may be lost along the way. This depends primarily on the amount of penalty imposed, which is determined by the shrinkage factor λ. Also, obtaining a high model performance commonly relies on the value of λ. However, chosen the best value of λ encounters an extensive amount of experiments, which can be subjected to the trial and error. Thus, the hard adoption of λ may increase the time-intensive and computational cost as well.

8. Conclusions

The SDN facilitates enterprises to upscale their network infrastructures with as little disruption as possible. The flexible and dynamic network architecture provided by the SDN is making its market continuously growing very quickly. Although the SDN paradigm can enhance the network management, the new architecture is prone to many security flaws and new threat vectors that can slow its widespread implementation. Hence, improvement of the IDS approaches is essential to boost SDN security. This study provided a comprehensive understanding of the problem by developing a new hybrid learning model based on the CNN algorithm. A new regularization method was adaptive to avoid the risk of overfitting and to enhance the efficiency of the CNN to predict whether a new observation of the data for each the model was not trained on it. Also, this paper solved one significant obstacle in the development of lightweight intrusion detection by training the DL models using a less number of features without causing a significant drop in the model performance. The experimental results show that the integrated CNN with the RF algorithm achieved higher performance and can enhance the accuracy of the intrusion detection. In our future work, we are planning to test the proposed model in a real SDN network environment and to explore how it can interact with the network attacks in real-time.

CRediT authorship contribution statement

Mahmoud Said ElSayed: Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation, Writing- Reviewing and Editing. **Nhien-An Le-Khac:** Visualization, Investigation, Software, Validation, Writing- Reviewing and Editing. **Marwan Ali Albahar:** Visualization, Investigation, Software, Validation, Writing- Reviewing and Editing. **Anca Jurcut:** Visualization, Investigation, Supervision, Software, Validation, Writing- Reviewing and Editing.

Declaration of competing interest

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

References

Abubakar, A., Pranggono, B., 2017. Machine learning based intrusion detection system for software defined networks. In: 2017 Seventh International Conference on Emerging Security Technologies. EST. IEEE, pp. 138–143.

Al-Qatf, M., Lasheng, Y., Al-Habib, M., Al-Sabahi, K., 2018. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access 6, 52843–52856.

Albahar, M.A., 2019. Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments. Secur. Commun. Netw. 2019, 1–9. http://dx.doi.org/10.1155/2019/8939041.

Cui, X., 2020. Directive-Based Data Partitioning and Pipelining and Auto-Tuning for High-Performance GPU Computing. Ph.D. thesis. Virginia Tech.

Depren, O., Topallar, M., Anarim, E., Ciliz, M.K., 2005. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Syst. Appl. 29 (4), 713–722.

Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., Ghorbani, A.A., 2016. Characterization of encrypted and vpn traffic using time-related. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy. ICISSP. pp. 407–414.

Elsayed, M.S., Le-Khac, N.-A., Dev, S., Jurcut, A.D., 2019. Machine-learning techniques for detecting attacks in SDN. In: 2019 IEEE 7th International Conference on Computer Science and Network Technology. ICCSNT. Dalian, China. IEEE.

Elsayed, M.S., Le-Khac, N.-A., Dev, S., Jurcut, A.D., 2020a. Ddosnet: A deep-learning model for detecting network attacks. In: 2020 IEEE 21st International Symposium on“ a World of Wireless, Mobile and Multimedia Networks”. WoWMoM. IEEE, pp. 391–396.

Elsayed, M.S., Le-Khac, N.-A., Jurcut, A.D., 2020b. Detecting abnormal traffic in large-scale networks. In: 2020 International Symposium on Networks, Computers and Communications. ISNCC. IEEE, pp. 1–7.

Elsayed, M.S., Le-Khac, N.-A., Jurcut, A.D., 2020c. InSDN: A novel SDN intrusion dataset. IEEE Access 8, 165263–165284.

Elsayed, M.S., Le-Khac, N.-A., Jurcut, A.D., 2021. Dealing with COVID-19 network traffic spikes [cybercrime and forensics]. IEEE Secur. Priv. 19 (1), 90–94.

Farahnakian, F., Heikkonen, J., 2018. A deep auto-encoder based approach for intrusion detection system. In: 2018 20th International Conference on Advanced Communication Technology. ICACT. IEEE, pp. 178–183.

Gao, X., Shan, C., Hu, C., Niu, Z., Liu, Z., 2019. An adaptive ensemble machine learning model for intrusion detection. IEEE Access 7, 82512–82521.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al., 2018. Recent advances in convolutional neural networks. Pattern Recognit. 77, 354–377.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

Isa, M.M., Mhamdi, L., 2020. Native SDN intrusion detection using machine learning. In: 2020 IEEE Eighth International Conference on Communications and Networking. ComNet. IEEE, pp. 1–7.

Jahromi, H.Z., Hines, A., Delanev, D.T., 2018. Towards application-aware networking: ML-based end-to-end application KPI/QoE metrics characterization in SDN. In: 2018 Tenth International Conference on Ubiquitous and Future Networks. ICUFN. IEEE, pp. 126–131.

Jan, S.U., Ahmed, S., Shakhov, V., Koo, I., 2019. Toward a lightweight intrusion detection system for the Internet of Things. IEEE Access 7, 42450–42471.

Javaid, A., Niyaz, Q., Sun, W., Alam, M., 2016. A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies. Formerly BIONETICS. pp. 21–26.

Karatas, G., Demir, O., Sahingoz, O.K., 2020. Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. IEEE Access 8, 32150–32162.

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity 2 (1), 1–22.

Kim, J., Kim, J., Kim, H., Shim, M., Choi, E., 2020. CNN-based network intrusion detection against denial-of-service attacks. Electronics 9 (6), 916.

Kim, J., Kim, H., et al., 2017. An effective intrusion detection classifier using long short-term memory with gradient descent optimization. In: 2017 International Conference on Platform Technology and Service. PlatCon. IEEE, pp. 1–6.

Klöti, R., Kotronis, V., Smith, P., 2013. OpenFlow: A security analysis. In: 2013 21st IEEE International Conference on Network Protocols. ICNP. IEEE, pp. 1–6.

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

Krishnan, P., Duttagupta, S., Achuthan, K., 2019. VARMAN: Multi-plane security framework for software defined networks. Comput. Commun. 148, 215–239.

Kumar, S., 2007. Survey of Current Network Intrusion Detection Techniques. Washington Univ. in St. Louis, pp. 1–18.

LeCun, Y., et al., 2015. LeNet-5, convolutional neural networks. URL: http://yann.lecun.com/exdb/lenet 20 (5) 14.

Li, J., Zhao, Z., Li, R., Zhang, H., 2018. Ai-based two-stage intrusion detection for software defined IoT networks. IEEE Internet Things J. 6 (2), 2093–2102.

Lin, W.-H., Lin, H.-C., Wang, P., Wu, B.-H., Tsai, J.-Y., 2018. Using convolutional neural networks to network intrusion detection for cyber threats. In: 2018 IEEE International Conference on Applied System Invention. ICASI. IEEE, pp. 1107–1110.

Liu, Y., Liu, S., Zhao, X., 2017. Intrusion detection algorithm based on convolutional neural network. DEStech Trans. Eng. Technol. Res. (iceta).

Luo, W., Liu, W., Gao, S., 2017. Remembering history with convolutional LSTM for anomaly detection. In: 2017 IEEE International Conference on Multimedia and Expo. ICME. IEEE, pp. 439–444.

Maaten, L.v.d., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (Nov), 2579–2605.

Malhotra, P., Vig, L., Shroff, G., Agarwal, P., 2015. Long short term memory networks for anomaly detection in time series. In: Proceedings, Vol. 89. Presses universitaires de Louvain, pp. 89–94.

Moustafa, N., Slay, J., 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference. MilCIS. IEEE, pp. 1–6.

Ng, A.Y., 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In: Twenty-First International Conference on Machine Learning - ICML '04. ACM Press, http://dx.doi.org/10.1145/1015330.1015435.

Oshiro, T.M., Perez, P.S., Baranauskas, J.A., 2012. How many trees in a random forest? In: International Workshop on Machine Learning and Data Mining in Pattern Recognition. Springer, pp. 154–168.

Pantiukhin, D.V., Voronkov, I.M., Nazarov, A.N., 2019. Intelligent methods for intrusion detection in local area networks. Expert Syst. 6, P. 7.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.-L., Chen, S.-C., Iyengar, S., 2018. A survey on deep learning: Algorithms, techniques, and applications. ACM Comput. Surv. 51 (5), 1–36.

Said Elsayed, M., Le-Khac, N.-A., Dev, S., Jurcut, A.D., 2020. Network anomaly detection using LSTM based autoencoder. In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks. pp. 37–45.

Santos, R., Souza, D., Santo, W., Ribeiro, A., Moreno, E., 2020. Machine learning algorithms to detect DDoS attacks in SDN. Concurr. Comput.: Pract. Exper. 32 (16), e5402.

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSp. pp. 108–116.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9.

Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M., 2018. Deep recurrent neural network for intrusion detection in sdn-based networks. In: 2018 4th IEEE Conference on Network Softwarization and Workshops. NetSoft. IEEE, pp. 202–206.

Verma, A., Ranga, V., 2018. Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. Procedia Comput. Sci. 125, 709–716.

Vigneswaran, K.R., Vinayakumar, R., Soman, K., Poornachandran, P., 2018. Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In: 2018 9th International Conference on Computing, Communication and Networking Technologies. ICCCNT. IEEE, pp. 1–6.

Wang, Y., Meng, W., Li, W., Li, J., Liu, W.-X., Xiang, Y., 2018. A fog-based privacy-preserving approach for distributed signature-based intrusion detection. J. Parallel Distrib. Comput. 122, 26–35.

Xiao, Y., Xing, C., Zhang, T., Zhao, Z., 2019. An intrusion detection model based on feature reduction and convolutional neural networks. IEEE Access 7, 42210–42219.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1492–1500.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C., 2018. Machine learning and deep learning methods for cybersecurity. IEEE access 6, 35365–35381.

Xu, H., Mueller, F., 2018. Machine learning enhanced real-time intrusion detection using timing information. In: International Workshop on Trustworthy & Real-Time Edge Computing for Cyber-Physical Systems.

Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K., 2018. Convolutional neural networks: An overview and application in radiology. Insights Imaging 9 (4), 611–629.

Yulianto, A., Sukarno, P., Suwastika, N.A., 2019. Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset. J. Phys. Conf. Ser. 1192 (1), 012018.

Zarpelão, B.B., Miani, R.S., Kawakani, C.T., de Alvarenga, S.C., 2017. A survey of intrusion detection in Internet of Things. J. Netw. Comput. Appl. 84, 25–37.

Zeiler, M.D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. Springer, pp. 818–833.

Mahmoud Said Elsayed received the B.E. degree in electronics and communication engineering from Zagazig University, Egypt in 2007, the M.E. degrees in Information Security from Nile University, Egypt, in 2018. He has worked for several years in the industry through Huawei and IBM Co. in area of Computer Network and Security. He is currently pursuing the Ph.D. degree at the School of Computer Science, UCD, Dublin, Ireland. His research interests involve computer networks, network security, deep learning, software-defined networks.

Dr. Nhien-An Le-Khac (M'06) Lecturer at the School of Computer Science (CS), University College Dublin (UCD), Ireland. He is currently the Program Director of UCD M.Sc. program in Forensic Computing and Cybercrime Investigation, an international program for the law enforcement officers specializing in cybercrime investigations. To date, more than 1000 students from 60 countries in 5 continents have graduated from this FCCI programme. He is also the co-founder of UCD-GNECB Postgraduate Certificate in fraud and e-crime investigation. He was a Research Fellow in Citibank, Ireland (Citi). He obtained his Ph.D. in Computer Science in 2006 at the Institut National Polytechnique de Grenoble (INPG), France. His research interest spans the area of Cybersecurity and Digital Forensics, Machine Learning for Security, Fraud and Criminal Detection, Cloud Security and Privacy and High Performance computing. Since 2013, he has collaborated on many research projects as a principal/co-PI/funded investigator. He has published more than 150 scientific papers in peer-reviewed journal and conferences in related research fields. He is an active chair as well as a reviewer for many key conferences and journals in related disciplines.

Dr. Marwan Ali Albahar received his B.S. in computer science from King Faisal University in 2011, Saudi Arabia, and his M.Sc. in computer science with honor in 2015 from Frostburg State University, USA. Dr. Albahar received 2018 his Ph.D. from the University of Eastern Finland. Dr. Albahar a senior Information Security, Privacy, and Risk Management Professional with a solid technical background and a highly analytical mind. He has been involved within the information security field for the last 3+. His main areas of research Computer Networks & Security, Cybersecurity, Artificial intelligence.

Dr. Anca Jurcut is an Assistant Professor in the School of Computer Science, University College Dublin (UCD), Ireland, since 2015. She received a B.Sc. in Computer Science and Mathematics from West University of Timisoara, Romania in2007 and a PhD in Security Engineering from the University of Limerick (UL), Ireland in 2013 funded by the Irish Research Council for Science Engineering and Technology. She worked as a postdoctoral researcher at UL as a member of the Data Communication Security Laboratory and as a Software Engineer in IBM in Dublin, Ireland in the area of data security and formal verification. Dr. Jurcut research interests include Security Protocols Design and Analysis, Automated Techniques for Formal Verification, Network Security, Attack Detection and Prevention Techniques, Security for the Internet of Things, and Applications of Blockchain for Security and Privacy. Dr. Jurcut has several key contributions in research focusing on detection and prevention techniques of attacks over networks, the design and analysis of security protocols, automated techniques for formal verification, and security for mobile edge computing (MEC). More Info: https://people.ucd.ie/anca.jurcut.