The Journal of Supercomputing (2021) 77:4618–4638
https://doi.org/10.1007/s11227-020-03447-z

Multi-level Gaussian mixture modeling for detection of malicious network traffic

Radhika Chapaneri[1] · **Seema Shah**[1]

Accepted: 1 October 2020 / Published online: 16 October 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract
Along with the growing network connectivity across the world, there is a substantial increase in malicious network traffic to exploit the vulnerabilities, thus hampering several organizations and end-users. Though signature-based and classification-based machine learning approaches can detect malicious network traffic, they cannot reliably detect unknown attacks. Several issues are yet unsolved using the existing approaches such as imbalanced training data, high false alarm rate, and lack of detection of unknown attacks. To address these issues, in this work, we propose a novel multi-level classification method that can accurately classify the network traffic into several classes and identify the novel attacks. The unsupervised Gaussian mixture modeling approach is used to learn the statistical characteristics of each traffic category, and an adaptive thresholding technique based on the interquartile range is used to identify any outlier. The proposed work is evaluated on the benchmark CICIDS2017 dataset that includes modern network traffic patterns. The results show a significant improvement relative to the state-of-the-art techniques for detecting unknown attacks and classifying multiple network traffic attacks.

Keywords Anomaly detection · Gaussian mixture model · Malicious network traffic · Multi-level classification

1 Introduction

The Internet's growth has increased drastically for performing various professional and personal tasks such as online shopping and banking. Moreover, the growth of modern devices is also an essential factor in the rise of Internet usage. However, along with this, there is also a significant surge of malicious users, and various

✉ Radhika Chapaneri
radhika.chapaneri@nmims.edu

Seema Shah
seema.shah@nmims.edu

[1] Department of Computer Engineering, MPSTME, NMIMS University, Mumbai, India

malicious attempts are made by exploiting vulnerabilities against individuals or organizations for stealing personal data, valuable information, or disrupting computational resources. The cyberattacks are increasing in complexity as well as the volume of attacks due to evolving technologies. As per the McAfee threat report [1], the top attack vectors in 2018–19 were malware, account hijacking, denial of service, and the most targeted sectors were individuals, healthcare, and finance.

The traditional job of security administrator monitoring the network data is becoming obsolete due to the use of automated tools via machine learning in determining malicious threat patterns in terabytes of data. Signature and anomaly-based approaches are the two most widely used approaches for malicious network traffic activity detection. The signature-based approach uses file hashes and custom written rules known as signatures to detect attacks; however, due to the sheer volume of intrusions or attacks, the signature-based techniques are not reasonable to detect various cyberattacks. Although these techniques have their merits, they require several hand-crafted rules for each type of attack, which requires a significant amount of manual work and regular updates of the signature database. The anomaly-based techniques *learn* the normal traffic behavior of the network and raise an alert whenever an anomalous behavior is detected. These techniques are crucial for defending networks and users by identifying various network attacks (mainly unknown) as various attacks grow. By identifying these attacks, the system can assist the network security administrators to take corresponding preventive or reactive measures.

The motivation of this research work is to adopt a novel approach to detect unknown attacks on the network. A fundamental assumption is that the benign or normal traffic data share common patterns, while the anomalous data in the form of the attack deviate from such patterns. We propose a method that uses unsupervised Gaussian mixture modeling (GMM) for benign and other network traffic classes and detects the anomalous traffic using adaptive thresholding based on the interquartile range. During the training phase, GMM models are learned for all known network traffic categories and evaluated using the metric of F1 score. The models are then arranged sequentially as per the descending F1 scores. In the testing phase, the test data's traffic category is obtained using a novel multi-level classification method that can also detect unknown attacks. The experimental results on the benchmark dataset show the feasibility of the proposed work relative to the state-of-the-art methods. The main contributions of this work are as follows:

(a) An unsupervised multi-level statistical-based approach using Gaussian mixtures with adaptive thresholding is proposed that can detect known as well as unknown network attacks.
(b) The pre-processing steps and visualization of the benchmark CICIDS2017 dataset are discussed in detail.
(c) A novel sample selection algorithm is proposed for selecting the representative samples of the dataset; this solves the imbalanced data problem present in the dataset.
(d) The performance of the proposed multi-level GMM approach is compared with existing machine learning and outlier detection methods for both binary and

multi-class classification. The results illustrate that the proposed approach is suitable for malicious activity detection from network traffic.

The rest of the paper is organized as follows: Sect. 2 discusses the previous studies related to this work. Section 3 details the proposed framework, along with a description of the CICIDS2017 dataset and the pre-processing steps. Section 4 presents the experimental setup and evaluation results, followed by the conclusion in Sect. 5.

2 Related work

A wide variety of vulnerabilities on the network or web can be exploited by performing various attacks such as the distributed denial of service, SQL injection, and cross-site scripting (XSS) [2], which can cause significant monetary loss. A botnet is a network of multiple machines using malware to execute malicious activities for corrupting and disrupting victim's resources, such as crashing websites. The botmaster can execute malicious activity controlling individual bot machines on the botnet simultaneously to complete a coordinated task to perform a massive scale attack.

In [3], the authors compared the performance of open-source signature-based systems Snort and Suricata; it was observed that since Suricata has multi-threaded architecture, it can process network traffic at a higher speed than Snort, but it consumes excessive computational resources. A significant disadvantage of such rule-based systems is that they cannot take any action against unknown malicious network traffic. For applying machine learning, a considerable amount of data for benign and various network attack classes is required; it is also not feasible to use public real-time network traffic due to privacy issues. Several benchmark datasets have been created to solve this problem, and they are used for evaluating the performance of the detection approaches. KDD/NSL KDD [4] is the most widely used dataset; however, a significant problem with this dataset is that it is now obsolete and does not provide an accurate representation of the current network attack scenarios. The UNSW-NB15 dataset [5] includes modern attacks as per the Common Vulnerabilities and Exposures (CVE) website [6], which publicly discloses and maintains the list of common vulnerabilities and exposures. The IXIA Perfectstorm tool, which has the ability to generate enterprise-level real-world traffic, was used to simulate nine categories of modern attacks in the UNSW-NB15 dataset. The following scenarios were performed: the first simulation for 16 hours with one attack/sec and second simulation for 15 hours with ten attacks/sec. Using modern tools such as Argus and Bro-IDS, the flow, basic, content, and time level features were extracted from the PCAP files. In [7], a comparison of various datasets is provided with the CICIDS2017 dataset based on eleven criteria such as complete network traffic and attack diversity, which are important for any dataset, and it was found that the CICIDS2017 dataset satisfies all eleven criteria. Due to this, the performance evaluation of the proposed work in this paper is done using the benchmark CICIDS2017 dataset.

Detecting malicious attacks can be considered as a classification problem, either binary or multi-class. In the literature, several machine learning approaches, including SVM [8], K-nearest neighbor (KNN) [9], artificial neural network [10], random forest (RF) [11], etc., have been proposed but are evaluated mostly on the KDD or NSL-KDD dataset. In [12], the author used three ANN architectures with the sigmoid activation function and backpropagation algorithm to detect DDoS attacks on TCP, UDP, or ICMP protocols. They were able to detect known attacks, but the accuracy of unknown attack detection was low. In [13] and [14], a comprehensive survey on recent machine learning-based network intrusion detection system (IDS) is presented by providing future directions.

In anomaly-based approaches, the pattern that deviates from a baseline model is called anomalous. Various anomaly-based approaches, such as nearest neighbor, clustering, statistical, and information-theoretic, along with their advantages and disadvantages, are discussed in [15]. In the Local Outlier Factor (LOF) approach, an outlier score is assigned to each data point based on distances from the local neighborhood. However, the LOF technique has a drawback in identifying outliers in datasets with varying densities. In clustering-based approaches, the data points that do not belong to any cluster are considered outliers. Since clustering aims to find clusters, these techniques are not optimized to find outliers. An intrusion prevention framework for mobile IoT devices was proposed in [16] by generating non-overlapping clusters and end-to-end security based on the blockchain architecture. In statistical-based approaches, the idea is that normal data instances occur in high probability regions, while anomalous data occurs in low probability regions. In [17], a GMM-based approach was used to detect flights with unusual data patterns relative to the normal flights.

To extract a better representation from the data, various techniques of deep learning such as convolutional neural network (CNN) [18], recurrent neural network (RNN) [19], long short-term memory (LSTM) [20] are proposed in the literature. In [19], RNN architectures were applied for the KDD dataset, and the results were compared with traditional machine learning models, and a 20 % increase in accuracy was obtained for both binary and multi-class classification but at the cost of increased processing time. To detect the web attacks in [2], the features are extracted from the raw HTTP request using a variant of autoencoder, namely stacked denoising autoencoder (SDAE). Since a single SDAE is not sufficient to recognize all malicious patterns, ensemble learning combines multiple SDAEs and reduces reconstruction errors. The Pelican intrusion detection framework was introduced in [21] and experimented on the UNSW-NB15 dataset, where it was observed that the performance of the neural network degrades by increasing the depth or layers. To resolve this issue, residual learning was implemented in Pelican to avoid the vanishing and exploding gradient problems. A combination of auto-encoders and deep neural network was proposed in [22] since the auto-encoders were shown to perform better than principal component analysis (PCA) for feature reduction; the resulting hybrid model led to an increase in the F1-score and a reduction in the false positive rate for the UNSW-NB15 dataset. A global anomaly threshold method was proposed in [23] for SCADA-based network data using unsupervised and ensemble models. Robust anomaly detection was proposed in [24] using cumulative error scoring,

percentile loss, and early stopping for auto-encoder modeling. The convolutional neural network was used in [25] for extracting the network features, and classification was performed using SVM considering that the abnormal network traffic is far away from the center of the hypersphere of the benign data; however, this evaluation was performed on the older KDD dataset.

It is worth noting that a considerable amount of data samples and a high processing time are required in these deep learning approaches. On the contrary, our proposed approach selects the representative samples from each category automatically, thus reducing the processing complexity. In this work, we propose a novel data preprocessing pipeline with a representative sample selection algorithm, GMM modeling for each traffic class, and a novel multi-level classification strategy with evaluation on the benchmark CICIDS2017 dataset.

3 Proposed work

Figure 1 shows the proposed framework for anomaly detection of network traffic using the multi-level Gaussian mixture modeling approach. It consists of several steps for data pre-processing followed by learning the Gaussian mixture model for each class. A multi-level classification technique is proposed based on the learned GMM models for classifying the input network traffic as either benign or a specific attack category. The multi-level technique can also detect any novel attack categories not already learned by the system.

3.1 Description of CICIDS2017 dataset

The proposed work is evaluated on the modern dataset CICIDS2017 [7] due to its realistic representation of benign traffic and diversity of attacks. This dataset is created by the Canadian Institute for Cybersecurity through a comprehensive testbed by creating two networks, one for the victim and second for the attack. The victim network consists of servers with various operating systems that are highly secured

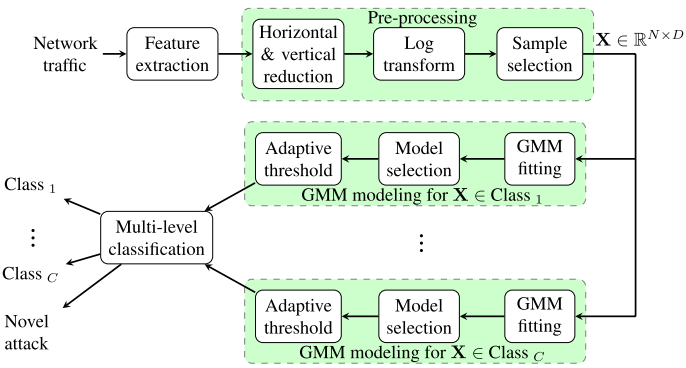


Fig. 1 Proposed framework

using firewalls and other network devices. The attack network consists of various devices, PCs with public IP's and necessary operating systems for executing the attack scenarios. The following prominent aspects motivated us to choose this dataset for evaluation:

- *Representation of real-world data*: The data is obtained from testbed created by using computers of the university having a recent and variety of operating systems (including Mac, Windows, Linux). Moreover, it also includes the HTTPS protocol in addition to the FTP, HTTP, and SSH protocols.
- *Up-to-date dataset*: The classes of attack in the CICIDS2017 dataset are align with the recent McAfee reports.
- *Labeled dataset*: The network flows are not only given the benign or attack labels but also the category of attack, which can be helpful to take adequate action.
- *Data availability*: Both raw data (PCAP - packet capture files) and CSV files are publicly available.

Table 1 summarizes the information about various attacks comprising the dataset. There are seven classes of attack, including DoS, WebAttacks, DDoS, Infiltration, etc., which are executed on separate days to have proper labeling. Multiple specific attacks are executed under each attack category using state-of-the-art tools. A detailed description of these attacks is given in [7]. The extraction of features is performed using CICFlowmeter, which extracts 79 statistical and time-related features from the PCAP files. Table 2 provides a description of the features of this dataset.

3.2 Data pre-processing

Performing thorough exploratory data analysis (EDA) and cleaning the dataset is considered to be an essential step; hence it is crucial to spend time exploring and cleaning data as it lays the foundation for the machine learning model. For data preprocessing, we have proposed steps for data cleaning, log transformation, and representative sample selection.

3.2.1 Horizontal and vertical data reduction

Since very few researchers have focused on using the CICIDS2017 dataset, a detailed analysis of this dataset can significantly contribute to the research community. For giving meaningful data to the machine learning model, we have performed horizontal and vertical reduction by removing unwanted features and redundant samples as follows:

- The FwdHeaderLength feature in the dataset that represents the total bytes of data flow in the forward direction is mentioned twice in the dataset; hence this error is corrected by deleting the repeated column.

Table 1 Summary of CICIDS2017 dataset network traffic

Traffic class (Pcap file size) day	Categories	Description	Tools	Security goal breached
Benign (10 GB) Monday	–	Normal Traffic	–	–
BruteForce (10 GB) Tuesday	FTP Patator, SSH Patator	Hit and try attack; Used for password cracking	Patator, Hydra, Metasploit	C
DoS (12 GB) Wednesday	Hulk, Slowhttptest, Slowloris, Golden Eye	Flooding with superfluous request to deny resources	Slowloris, GoldenEye, Heartleech	A
WebAttacks (7.7 GB) Thursday (Mor.)	SQL Injection, Bruteforce XSS	Exploits web vulnerabilities	Damn vulnerable Webapp	I
Infiltration (103 MB) Thursday (Aft.)	–	Exploits vulnerable software	Metasploit	C
Bot (8.2 GB) Friday (Mor.)	–	Multiple Internet connected devices to perform malicious tasks	Grum, Widigo, Storm, Ares	C, A
Portscan (92.7 MB) Friday (Aft.)	SYN scan, ACK scan, FIN scan, UDP scan	Reconnaissance; learn critical information about connected devices	Nmap	C
DDoS (97.1 MB) Friday (Aft.)	–	Multiple devices used to flood the bandwidth or resources of a victim	LOIC, HOIC	A

Security Goals – C: Confidentiality, A: Availability, I: Integrity

Table 2 Features of the CICIDS2017 dataset

#	Feature	Description
1	Dest. Port	Destination Port
2	Flow Duration	Duration of the flow in microseconds
3–4	Total Fwd/Bwd packets	Total packets in the forward/backward dir.
5–6	Total length of Fwd/Bwd packets	Total size of packet in forward/backward dir.
7–14	Fwd/Bwd packet length	Min, Max, Mean, Std size of packet in forward/backward dir.
15–16	Flow Bytes/Packets/s	Number of flow bytes/packets per sec.
17–30	Flow IAT/Fwd IAT / Bwd IAT	Min, Max, Mean, Std and total time between packets in flow/fwd/bwd dir.
31–34	PSH/URG Flag	# times PSH / URG flag set in packets in fwd/bwd direction (0 for UDP)
35–36	Fwd/Bwd header length	Total header bytes in forward direction
37–38	Fwd/Bwd Packet/s	Number of forward packets per second
39–43	Packet length	Min, Max, Mean, Std, Variance
44–51	Flag counts	Number of packets with PSH, SYN, FIN, RST, ACK, URG, CWE, ECE
52	Down/Up ratio	Download and upload ratio
53	Average packet size	Average size of the packet
54–55	Avg Fwd/Bwd seg size	Size observed in Fwd/Bwd direction
56	Fwd header length	Length of header for forward packet
57–62	Fwd/Bwd Avg Bytes/Packets/Bulk	Average number of bytes/ packets/ bulk rate in the forward/backward direction
63–66	SubFlow Fwd/Bwd Bytes	Average number of subflow bytes in forward and backward direction
67–68	Init_Win_Bytes_Forward/Backward	Total number of bytes in TCP initial window size in forward/backward dir.
69	Act_data_packet_forward	Count of packets with at least 1 byte of TCP data payload in forward dir.
70	Min_seg_size_forward	Minimum segment size in forward dir.
71–74	Active time	Min, Mean, Max, Std time a flow was active before becoming idle
75–78	Idle time	Min, Mean, Max, Std time a flow was active before becoming active
79	Label	Benign/Attack category

- The data samples having NaN values for the features FlowBytes/s and FlowPackets/s are removed.
- After analyzing the dataset, the following features having all zero values are dropped: (a) BwdPshFlags, (b) FwdURGFlags, (c) BwdURGFlags, (d) CWEFlag-Count, (e) FwdAvgBytesBulk, (f) FwdAvgPacketsBulk, (g) FwdAvgBulkRate, (h) BwdAvgBulkRate, (i) BwdAvgPacketBulk, and (j) BwdAvgBulkrate.
- The packet length variance feature is removed due to the presence of packet length std deviation feature; thus, the redundancy is avoided.
- IdleMean, IdleStd, IdleMax, IdleMin features are removed as they have a very high standard deviation; thus, the inconsistent features across the dataset are avoided.
- The irrelevant data samples having negative values for the following features are removed: (a) FlowDuration, (b) FlowBytes, (c) FlowPackets/s, (d) FlowIATMean, (e) BwdIAT, (f) FwdHeaderLength and g) BwdHeaderLength. These features are required to have non-negative values, and thus, these irrelevant samples are removed.
- The Infiltration class of attack is not considered as the number of samples is too small to train the learning model.

This data cleaning process is a time-consuming but essential step for further data processing of building a powerful predictive model.

3.2.2 Feature transformation

Several feature values of this dataset span varying orders of magnitude, for example, the range of FlowDuration is from 900 to 1,600,000. All feature values are transformed into the log domain using Eq. (1) thus reducing their scale and aiding the learning model to work with smaller feature values. An illustration of feature transformation is shown in Table 3.

$$x \leftarrow \log(x + 1) \tag{1}$$

3.2.3 Sample selection

Considering the distribution of attacks in the CICIDS2017 dataset as shown in Table 4, there are more benign samples relative to the malicious samples. Also, the

Table 3 Log transformation of features

Original value		Transformed value	
Flow duration	FlowIAT mean	Flow duration	FlowIAT mean
1,581,930	60,843.46	14.27	11.01
2,346,262	111,726.76	14.66	11.62
3,266,983	148,499.00	14.99	11.90
915	457.50	6.82	6.12
9,749	2,437.25	9.10	7.79

distribution of attacks is highly imbalanced, for example, the DoS attack is almost half of all attacks and the Portscan attack is one-third of all attacks as they cause too many packets flows during the attack.

Algorithm 1 Sample Selection

Input: $\mathbf{X}^{'}(N_i^{'} \times D)$, distance threshold λ
Output: Representative selected samples $\mathbf{X}\,(N_i \times D)$

1: $\mathbf{C}_1 = \mathbf{X}^{'}(1,:)$ ▷ first cluster centroid
2: $N_i = 1$
3: **for** $j = 2, \ldots, N_i^{'}$ **do**
4: $\quad \mathbf{S} = \mathbf{X}^{'}(j,:)$ ▷ read the next sample
5: $\quad$ **if** $\exists \mathbf{C}_p \in \{\mathbf{C}_k : 1 \leq k \leq N_i\},\ d(\mathbf{S}, \mathbf{C}_p) < \lambda$ **then**
6: $\quad\quad$ put $\mathbf{S}$ in p^{th} cluster
7: $\quad\quad$ update the cluster centroid $\mathbf{C}_p$
8: $\quad$ **else**
9: $\quad\quad N_i = N_i + 1$ ▷ update number of clusters
10: $\quad\quad \mathbf{C}_{N_i} = \mathbf{S}$ ▷ create a new cluster
11: $\mathbf{X} = [\mathbf{C}_1, \ldots, \mathbf{C}_{N_i}]^{\top}$ ▷ representative samples

To select the most representative samples for each category, we propose a sample selection algorithm given by Algorithm 1 taking input as $\mathbf{X}'$ consisting of N' data samples resulting from horizontal reduction with D features and a distance threshold λ. The threshold λ is a hyper-parameter that can be obtained as the median of all distance pairs for a particular category. The first cluster centroid $\mathbf{C}_1$ is chosen as the first instance of the dataset. Each subsequent sample of the dataset is assigned to $\mathbf{S}$ whose distance d is computed from the clusters generated so far in line 5. If the distance d is less than the threshold λ with respect to the p^{th} cluster, then this sample $\mathbf{S}$ is included in the p^{th} cluster and the centroid of cluster $\mathbf{C}_p$ is updated in a manner similar to the method adopted in K-means clustering. Otherwise, this sample defines a new cluster $\mathbf{C}_{N_i}$ where N_i is the total number of resulting clusters. The representative samples $\mathbf{X} \in \mathbb{R}^{N_i \times D}$ are thus the cluster centroids $\mathbf{C}_1, \ldots, \mathbf{C}_{N_i}$. This algorithm is

Table 4 Representative samples of CICIDS 2017 dataset after sample selection

Class	# Original samples	# Samples after horizontal and vertical reduction	# Representative samples after sample selection
Benign	529,918	231,187	4195
Bot	1966	1956	1956
Bruteforce	13,835	13,826	3998
DDos	128,027	128,006	4415
DoS	252,672	251,555	4,954
Portscan	158,930	158,797	3081
WebAttacks	2180	2180	2180
Total	**1,087,528**	**787,507**	**24,779**

The bold row refers to the total number of data samples

repeated for each category of data, i.e. Benign, DoS, etc., Table 4 shows the number of samples before sample selection as well as the number of representative samples after the sample selection algorithm.

3.3 Data visualization

It is infeasible to visually explore the data since the network traffic features are of high-dimensional. Hence, t-SNE (t-distributed stochastic neighbor embedding) [26] is used for projecting the high-dimensional features to a low-dimensional subspace such that the Kullback–Leibler (KL) divergence between their corresponding distributions is minimized in a non-linear manner. The t-SNE scatter plot of CICIDS2017 dataset projected onto two dimensions without and with pre-processing is shown in Fig. 2a ,b, respectively. It can be observed that there is considerable overlap between various traffic classes without pre-processing, whereas the proposed pre-processing results in almost distinct clusters and thus discriminative features which can reduce the effort of learning models.

The boxplot of two features across the traffic categories is shown in Fig. 3, where the advantage of proposed data pre-processing steps can be observed as the feature representation after pre-processing is more discriminative for machine learning models to learn.

3.4 GMM modeling with adaptive thresholding

For each category of data samples, Gaussian Mixture Models (GMM) are learned to model the probability distribution of the features using which anomalous or malicious data samples of network traffic can be identified.

3.4.1 GMM training

Since the data cannot be represented appropriately with only one Gaussian, K mixtures of Gaussians are considered given by Eq. (2) with model parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ indicating the weight, mean and covariance of the k^{th} latent

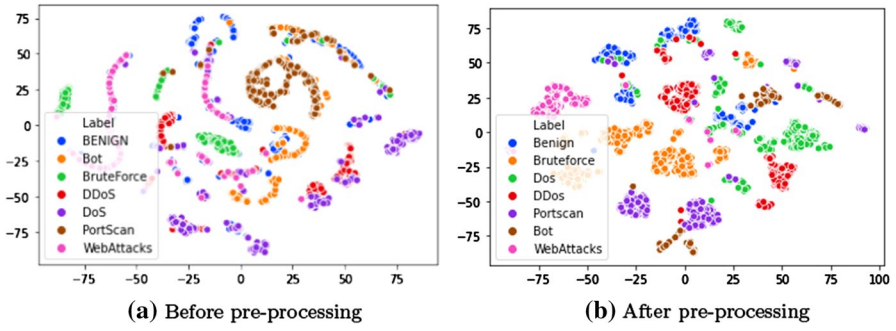


Fig. 2 t-SNE visualization of the CICIDS 2017 dataset

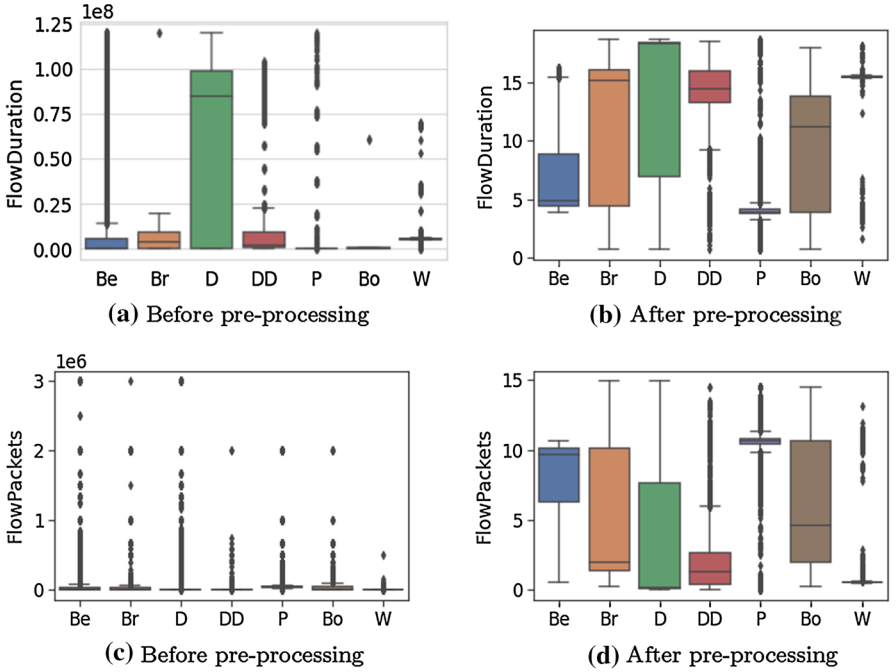


Fig. 3 Boxplot visualization for the impact of pre-processing for FlowDuration (top) and FlowPackets (bottom) features. Here, Be: Benign, Br: Bruteforce, D: DoS, DD:DDoS, P: Portscan, Bo: Bot, W: Web-Attacks

variable (or mixture), and $\mathbf{x} \in \mathbb{R}^D$ is the data sample having D features. In GMM, $\mathbf{z}$ is a K-dimensional binary random variable having a one-hot representation with $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$.

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{2}$$

The goal of GMM is to estimate the model parameters using maximum likelihood estimation for which the EM (Expectation Maximization) algorithm is used. The log-likelihood function is given by Eq. (3) assuming that the data points $\mathbf{x}_n$ are sampled independently.

$$LL = \ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N_i} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \right\} \tag{3}$$

Algorithm 2 Training - GMM based Anomaly Detection

Input: $\mathbf{X}\,(N_i \times D)$, K (number of clusters)
Output: $profile : \{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, Q_1, Q_3\}$

1: Initialize $\pi_k = \frac{1}{K}$, $\boldsymbol{\mu}_k$ as K-means center $(1 \times D)$, $\boldsymbol{\Sigma}_k$ as covariance of the data $(D \times D)$, $\forall k = 1, \ldots, K$, and $LL^{(0)} = \inf$
2: **E-step**: Compute the posterior probabilities γ_{nk}, $\forall k = 1, \ldots, K$
3: **M-step**: Update the GMM parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, $\forall k = 1, \ldots, K$
4: Compute the log likelihood $LL = \ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. If LL not converged, return to Step 3.
5: Obtain log likelihood *scores* for every training sample.
6: Compute $Q_1 = 25$ percentile of *scores* and $Q_3 = 75$ percentile of *scores*.

The training algorithm of GMM modeling is given by Algorithm 2, which is executed for each class of the dataset. The model parameters are initialized in line 1 and the E-step computes the *responsibility* that the component k takes for explaining the observation $\mathbf{x}_n$ as given by Eq. (4), which is obtained as the posterior probability using Bayes' theorem, where π_k is the corresponding prior probability. The M-step is computed in line 3, where using the current responsibilities, the parameters are updated as given by Eq. (5) obtained by setting the derivatives of the log-likelihood function to zero with respect to each parameter. Here, N_k denotes the number of data samples in the k^{th} cluster. The log-likelihood is recomputed in line 4 using the updated parameters and the E and M steps are repeated till convergence of the log-likelihood.

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}\left(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \tag{4}$$

$$\begin{aligned} \boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n, \\ \boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\top}, \\ \pi_k^{\text{new}} &= \frac{N_k}{N}, \quad N_k = \sum_{n=1}^{N} \gamma_{nk} \end{aligned} \tag{5}$$

After the EM algorithm converges, the log likelihood value is computed for each training sample and stored in *scores* from which the first (Q_1) and third (Q_3) quartiles are computed. These statistics are used during the testing phase for adaptive thresholding. An illustration of one-dimensional Gaussian distribution is shown in Fig. 4 showing the two quartiles Q_1 and Q_3 as well as the unfilled region of interquartile range $IQR = Q_3 - Q_1$. Any value lower than $Q_1 - 1.5 \times IQR$ or more than $Q_3 + 1.5 \times IQR$ will have a very less probability hence can be considered to be anomalous with respect to the given distribution.

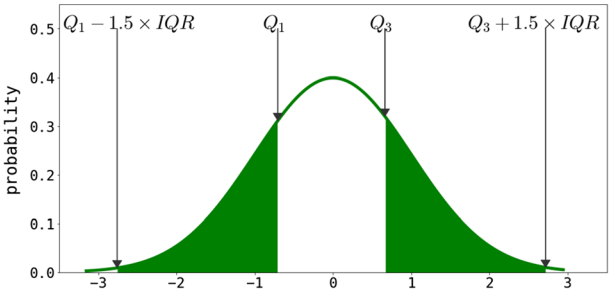


Fig. 4 Illustration of IQR for 1D Gaussian distribution

3.4.2 Model selection

Since the choice of K in GMM is a hyper-parameter, the optimal value of K is chosen using model selection with BIC (Bayesian Information Criterion) measure given by Eq. (6), where for any model $\mathcal{M}_i$, $LL(\mathcal{M}_i)$ represents the log-likelihood of data with respect to $\mathcal{M}_i$, N is the number of data samples and $t(\mathcal{M}_i)$ is the number of model parameters.

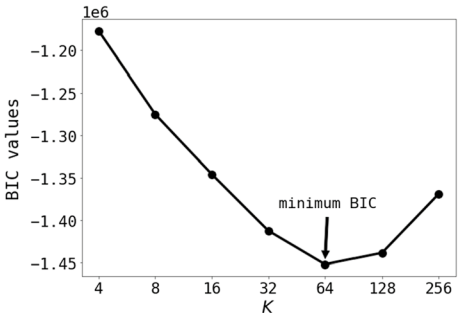
$$BIC(\mathcal{M}_i) = -2 \times LL(\mathcal{M}_i) + \log(N) \times t(\mathcal{M}_i) \tag{6}$$

For each class, various values of $K = \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$ is used for training the GMM model, and the model resulting in the lowest BIC value is retained as the optimal model since the BIC measure penalizes models with a high number of clusters. An illustration of the model selection using BIC is shown in Fig. 5 for the benign category.

3.4.3 Testing phase

In the testing phase as shown in Algorithm 3, data samples of all classes are combined together resulting in $\mathbf{X}_t \in \mathbb{R}^{N \times D}$ and the ground truth for each test sample is set as follows: $y_{true} = 1$ if the sample belongs to the desired category, else $y_{true} = 0$.

Fig. 5 BIC plot for Benign class



The IQR thresholding technique is used to determine whether the data sample belongs to a particular class or not.

Algorithm 3 Testing - GMM based Anomaly Detection

Input: $\mathbf{X}_t\ (N \times D)$ (all samples),
$profile : \{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, Q_1, Q_3\}$
Output: Decision label for each sample

1: Compute $IQR = Q_3 - Q_1$
2: Compute $lb = Q1 - 1.5 \times IQR$ and $ub = Q3 + 1.5 \times IQR$
3: **for** each sample $\mathbf{x}_i \in \{\mathbf{X}_t\}_{i=1}^{N}$ **do**
4: Compute the log likelihood score $score(i) = \ln p(\mathbf{x}_i|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
5: **if** $score(i) < lb$ or $score(i) > ub$ **then**
6: $label(i) =$ 'Anomaly'
7: **else**
8: $label(i) =$ 'Benign'

For each test sample, the log-likelihood score is computed and compared with the IQR, calculated as $IQR = Q_3 - Q_1$. The IQR criterion is useful since extreme values has less influence on it as it limits the range to middle 50% of the score values. The values for $Q_1 - 1.5 \times IQR$ and $Q_3 + 1.5 \times IQR$ are the thresholds that are used to determine the outlier values. The decision is taken in lines 4–8 using the IQR threshold to assign Benign or Anomaly label; to the specific data sample, assuming that the GMM profile of the Benign class profile is given as the input, but this applies in general to any class.

3.5 Multi-level classification

Most existing research focuses on either binary classification of network traffic (Benign or Anomalous) or multi-class classification (Benign, DoS, Backdoor, etc.). To detect the specific class of network traffic data samples as well as to detect novel attacks (zero-day attack), a multi-level classification approach based on GMM is proposed as shown in Fig. 6. Based on the trained GMM models for each class of network traffic, these models are evaluated against the validation set samples to determine their individual F1-scores given by Eq. (7). Here, *TP*, *TN*,

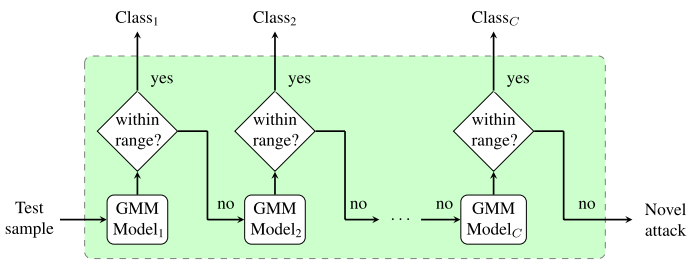


Fig. 6 Proposed multi-level classification framework

FP and *FN* denote the number of true positives, true negatives, false positives, and false negatives, respectively. A higher F1-score implies lower false positives as well as false negatives and hence is a better evaluation metric compared to the accuracy.

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + FP + TN + FN}, \\ Precision &= \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \\ F1score &= \frac{2 \times Precision \times Recall}{Precision + Recall} \end{aligned} \tag{7}$$

The representative samples' dataset obtained for each class after sample selection is split into $60\%, 20\%, 20\%$ train, validation, and test set [28]. Table 5 shows the performance of individual GMM models trained for each class of the attacks of the CICIDS2017 dataset. The precision, recall, and F1-score values are obtained on the validation sample set. It can be observed that each trained GMM model can identify the specific class samples with higher F1-scores. These F1-scores are sorted in the descending order to determine the multi-level ordering of classes as follows: **DDoS → Benign → Bruteforce → Bot → DoS → WebAttacks → Portscan**.

For the simplicity of discussion, consider three classes, namely Bruteforce, DoS, and WebAttacks. Their trained GMM models are arranged in this order of decreasing F1-scores. For each test data sample, first, the Bruteforce GMM model will apply the IQR threshold criteria to determine if the log-likelihood *score* is within its IQR range; if yes, then this sample is labeled as Bruteforce, otherwise, the procedure is repeated for DoS, and WebAttacks GMM trained models. Suppose the log-likelihood score of the test sample does not satisfy the IQR criteria of all three GMM models. In that case, this test sample is denoted as a novel attack or a zero-day attack sample. The proposed multi-level approach can thus determine any novel attacks. This strategy can help security administrators take appropriate action based on the type of attack or novel attacks and take further preventive steps.

Table 5 GMM model evaluation per class

Class	*K*	Precision	Recall	F1-score
Benign	64	0.98	0.98	0.98
Bot	32	1.00	0.91	0.95
BruteForce	64	1.00	0.93	0.96
DDoS	64	1.00	0.98	0.99
DoS	128	1.00	0.88	0.93
Portscan	64	0.61	0.93	0.73
WebAttacks	64	1.00	0.83	0.90

4 Experimental results

The proposed work's objective is to effectively classify the network traffic data as either Benign or Anomalous and the type of the attack, including novel attacks. Various experiments are conducted to validate the working of the proposed work with respect to binary and multi-class classification and compared with several state-of-the-art existing works detailed as follows. In [27], principal component analysis (PCA) and auto-encoding techniques are applied to reduce the feature dimensionality of traffic data and four methods, namely, local outlier factor (LOF), one-class support vector machine (OCSVM), isolation forest (IF), and robust covariance (RC), are evaluated for network anomaly detection. In [28], deep belief network (DBN) along with an ensemble SVM is applied in a distributed framework using Apache Spark for large-scale network intrusion detection. In [29], Portscan attempts are detected for the CICIDS2017 dataset using the SVM classifier. An in-depth analysis of the CICIDS2017 dataset is conducted in [30] where the following models are evaluated for multi-class classification: Adaboost, Multi-layer Perceptron (MLP), Naive Bayes (NB), and Quadratic Discriminant Analysis (QDA). It is observed in [30] that DoS and DDoS attack patterns being similar to benign traffic are difficult to distinguish using the traditional machine learning models. Synthetic minority oversampling technique (SMOTE) was used in [31] to address the dataset imbalance issue, and the Adaboost classifier was used for network intrusion detection. All experiments in this work are implemented on a Windows machine with Intel Core i5-7200U 2.5GHz processor having 8GB RAM Nvidia Geforce 940MX GPU.

4.1 Binary classification

For binary classification, the samples of the dataset are classified as either belonging to the Benign class or the Anomalous class that comprises of all attacks. The proposed work is compared against traditional classifiers and various well-known anomaly detection algorithms (LOF, OCSVM, DBN, etc.) from [27–29]. Table 6 shows the results of binary classification, where it can be observed that the

Table 6 Evaluation result of binary classification

Approach	Precision	Recall	F1score
LOF [27]	0.76	0.89	0.82
OCSVM [27]	0.70	0.80	0.75
IF [27]	0.87	0.75	0.81
RC [27]	0.87	0.72	0.79
DBN [28]	0.90	0.50	0.65
Ensemble [28]	0.89	0.92	0.91
MLP [28]	0.90	0.94	0.92
SVM [29]	0.80	0.70	0.75
Proposed Work	**0.97**	**0.95**	**0.96**

Bold highlights the results obtained with the proposed work

proposed work outperforms the existing techniques for classifying the samples as either Benign or Anomalous.

4.2 Multi-class classification

For the multi-class classification of the test data samples, the proposed multi-level approach is compared with the existing techniques. Table 7a shows the results of individual GMM models for the test dataset considering all known network traffic classes, where it can be observed that test samples can be classified effectively with the average F1-score of 0.95.

To determine the performance of the proposed system to detect unknown attacks, the following classes were used in the multi-level classification system: Benign, Bot, BruteForce, DDoS, and DoS. Portscan and WebAttacks are considered to be unknown attacks, i.e., these classes are not considered during multi-level classification. Thus, only five GMM models are used in the testing phase, and for each test data sample, it is assigned the appropriate label if it belongs to the probability distribution of the specific GMM, else it is assigned an Unknown label if it is an outlier to any of the existing GMM models. Table 7b illustrates the performance of this scenario where we observe that the proposed multi-level system can identify the unknown attacks with a high F1-score.

Further, the proposed work is compared with existing multi-class classifiers in the literature and the results are shown in Table 8. The binary as well as multi-class classification results are pictorially illustrated in Fig. 7. The results show that the proposed approach achieves the highest F1-score compared to existing approaches.

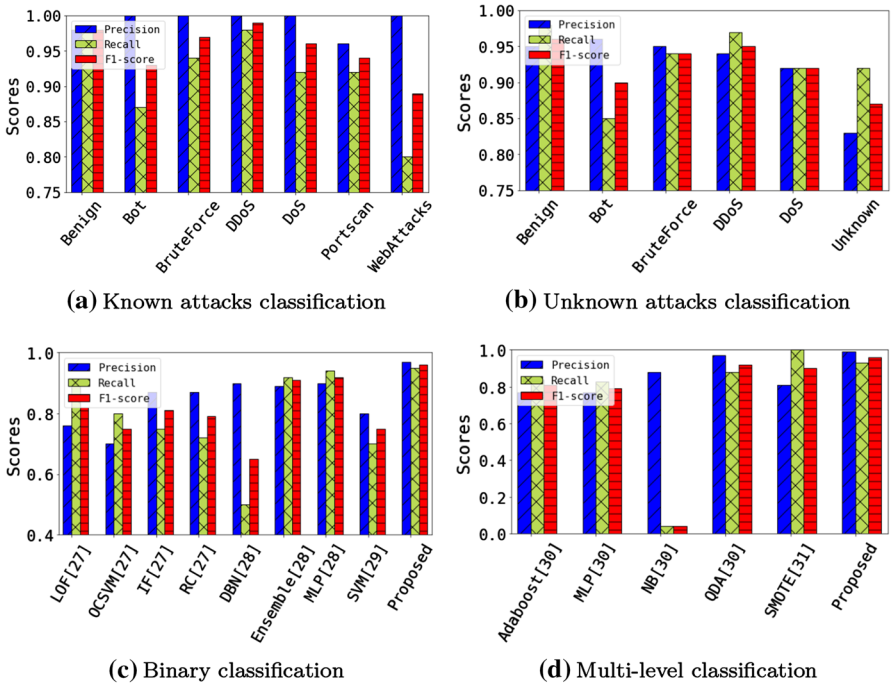
Table 7 Evaluation results of multi-class classification (Pr: Precision, Rc: Recall, F1: F1-score)

Class	Pr	Rc	F1
(a) Known attacks classification			
Benign	0.98	0.98	0.98
Bot	1.00	0.87	0.93
BruteForce	1.00	0.94	0.97
DDoS	1.00	0.98	0.99
DoS	1.00	0.92	0.96
Portscan	0.96	0.92	0.94
WebAttacks	1.00	0.80	0.89
(b) Unknown attacks classification			
Benign	0.95	0.98	0.96
Bot	0.96	0.85	0.90
BruteForce	0.95	0.94	0.94
DDoS	0.94	0.97	0.95
DoS	0.92	0.92	0.92
Unknown	0.83	0.92	0.87

Table 8 Performance comparison with existing work

Approach	Precision	Recall	F1-score
Adaboost [30]	0.77	0.84	0.81
MLP [30]	0.77	0.83	0.79
NB [30]	0.88	0.04	0.04
QDA [30]	0.97	0.88	0.92
Adaboost with SMOTE [31]	0.81	1.00	0.90
Proposed work	**0.99**	**0.93**	**0.96**

Bold highlights the results obtained with the proposed work



(a) Known attacks classification

(b) Unknown attacks classification

(c) Binary classification

(d) Multi-level classification

Fig. 7 Performance evaluation of proposed work

5 Conclusion and future scope

In this work, a novel multi-level classification method is proposed to accurately classify the network traffic into benign as well as malicious traffic categories such as DDoS, Portscan, and WebAttacks. Using the statistical Gaussian mixture modeling approach, the pattern of each category is learned to identify any potential outliers based on the inter-quartile range. Any data sample exceeding the threshold of the log probability scores is treated as an outlier for a specific traffic category. The GMM models are cascaded serially for the multi-level classification based on the

descending F1-scores of individual models. Therefore, this approach is capable of identifying any unknown attack or zero-day attack if the test data does not belong to the profiles of any learned GMM distribution. The proposed work is evaluated on the benchmark CICIDS2017 dataset and is shown to outperform the existing work in the literature. One limitation of this work is that the GMM profile for Benign and various attack classes need to be updated frequently due to changing network attack patterns. Recently, deep neural networks have been found to be effective in many applications, including malicious activity detection; however, the training of deep neural networks requires a significant amount of data. Generative adversarial networks (GAN) are also found to be effective in generating new data samples as per the probability distribution of existing data. Due to privacy and security concerns for network traffic data collection, we intend to explore the use of GAN for generating more data samples for specific rare attack classes for further work.

References

1. McAfee Labs (2019) McAfee Labs threats report, pp. 1–60. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf. Accessed 14 May 2020
2. Truong D, Tran D, Nguyen L, Mac H, Tran H, Bui T (2019) Detecting web attacks using stacked denoising autoencoder and ensemble learning methods. In: ACM Int. Conf. Proceeding Series pp 267–272. https://doi.org/10.1145/3368926.3369715
3. Shah S, Issac B (2018) Performance comparison of intrusion detection systems and application of machine learning to Snort system. Future Generation Computer Systems 80:157–170. https://doi.org/10.1016/j.future.2017.10.016
4. Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) A detailed analysis of the KDD CUP 99 dataset. In: Proc. of IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA), Ottawa, Canada. https://doi.org/10.1109/CISDA.2009.5356528
5. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems. In: Proc. of IEEE Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, pp 10–15. https://doi.org/10.1109/MilCIS.2015.7348942
6. CVE Identifiers https://cve.mitre.org/cve/ Accessed 4 Jan 2020
7. Sharafaldin I, Gharib A, Lashkari A, Ghorbani A (2017) Towards a reliable intrusion detection benchmark dataset. Softw Netw 1:177–200. https://doi.org/10.13052/jsn2445-9739.2017.009
8. Kuang F, Xu W, Zhang S (2014) A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl Soft Comput 18:178–184. https://doi.org/10.1016/j.asoc.2014.01.028
9. Li W, Yi P, Wu Y, Pan L, Li J (2014) A new intrusion detection system based on KNN classification algorithm in wireless sensor network. J Electr Computer Eng 2014:240217. https://doi.org/10.1155/2014/240217
10. Ingre B, Yadav A (2015) Performance analysis of NSL-KDD dataset using ANN. In: IEEE Intl. Conf. on Signal Processing and Communication Engineering Systems pp. 92–96. https://doi.org/10.1109/SPACES.2015.7058223
11. Farnaaz N, Jabbar M (2016) Random forest modeling for network intrusion detection system. Procedia Comput Sci 89:213–217
12. Saied A, Overill R, Radzik T (2016) Detection of known and unknown DDoS attacks using artificial neural networks. Neurocomputing 172:385–393. https://doi.org/10.1016/j.neucom.2015.04.101
13. Bhuyan M, Bhattacharyya D, Kalita J (2013) Network anomaly detection: methods, systems and tools. IEEE Commun Surv Tutor 16:303–336. https://doi.org/10.1109/SURV.2013.052213.00046
14. Chapaneri R, Shah S (2018) A comprehensive survey of machine learning-based network intrusion detection. Smart Intell Comput Appl 104:345–356. https://doi.org/10.1007/978-981-13-1921-1_35
15. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41:1–58. https://doi.org/10.1145/1541880.1541882

16. Haseeb K, Islam N, Almogren A, Ud Din I (2019) Intrusion prevention framework for secure routing in WSN-based mobile Internet of things. IEEE Access 7:185496–185505. https://doi.org/10.1109/ACCESS.2019.2960633
17. Li L, Hansman R, Palacios R, Welsch R (2016) Anomaly detection via a Gaussian mixture model for flight operation and safety monitoring. Transp Res Part C Emerg Technol 64:45–57. https://doi.org/10.1016/j.trc.2016.01.007
18. Chapaneri R, Shah S (2019) Detection of malicious network traffic using convolutional neural networks. In: IEEE 10th Intl. Conf. on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India. https://doi.org/10.1109/ICCCNT45670.2019.8944814
19. Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5:21954–21961. https://doi.org/10.1109/ACCESS.2017.2762418
20. Kim J, Thu H, Kim H (2017) Long short term memory recurrent neural network classifier for intrusion detection. In: IEEE Intl. Conf. Platform Technology and Service, South Korea. https://doi.org/10.1109/PlatCon.2016.7456805
21. Wu P, Guo H, Moustafa N (2020) Pelican: a deep residual network for network intrusion detection. In: 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). https://doi.org/10.1109/DSN-W50199.2020.00018
22. Dutta V, Choraś M, Kozik R, Pawlicki M (2020) Hybrid model for improving the classification effectiveness of network intrusion detection. In: Intl. Conference on Complex, Intelligent, and Software Intensive Systems, Springer, Berlin https://doi.org/10.1007/978-3-030-57805-3_38
23. Almalawi A, Fahad A, Tari Z, Khan A, Alzahrani N, Bakhsh S, Alassafi M, Alshdadi A, Qaiyum S (2020) Add-on anomaly threshold technique for improving unsupervised intrusion detection on SCADA data. Electronics 9:1017. https://doi.org/10.3390/electronics9061017
24. Merrill N, Eskandarian A (2020) Modified autoencoder training and scoring for robust unsupervised anomaly detection in deep learning. IEEE Access. https://doi.org/10.1109/ACCESS.2020.2997327
25. Chen X, Cao C, Mai J (2020) Network anomaly detection based on deep support vector data description. In: 5th IEEE International Conference on Big Data Analytics (ICBDA). https://doi.org/10.1109/ICBDA49040.2020.9101325
26. Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
27. Pérez D, Alonso S, Morán A, Prada M, Fuertes J, Dománguez M (2019) Comparison of network intrusion detection performance using feature representation. In: Intl. Conf. on Engineering Applications of Neural Networks pp 463–475. https://doi.org/10.1007/978-3-030-20257-6_40
28. Marir N, Wang H, Feng G, Li B, Jia M (2018) Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2875045
29. Aksu D, Aydin M (2018) Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In: IEEE Intl. Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT) Turkey pp 77–80. https://doi.org/10.1109/IBIGDELFT.2018.8625370
30. Sharafaldin I, Habibi L, Ghorbani A (2018) A detailed analysis of the CICIDS2017 data set. In: Intl. Conf. on Information Systems Security and Privacy, Springer Cham.https://doi.org/10.1007/978-3-030-25109-3_9
31. Yulianto A, Sukarno P, Suwastika N (2019) Improving AdaBoost-based intrusion detection system performance on CICIDS 2017 dataset. In: Journal of Physics: Conference Series, vol 1192, no. 1. https://doi.org/10.1088/1742-6596/1192/1/012018

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.