Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

Cyber threat intelligence using PCA-DNN model to detect abnormal network behavior

Mohammad Al-Fawa'reh [a,*], Mustafa Al-Fayoumi [b], Shadi Nashwan [c], Salam Fraihat [d]

[a] College of Information Technology and Computer Science, Yarmouk University, Jordan
[b] King Hussein School of Computing Sciences, Princess Sumaya University for Technology (PSUT), Jordan
[c] College of Computer and Information Sciences, Jouf University, Saudi Arabia
[d] Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, UAE

ARTICLE INFO

Article history:
Received 13 July 2021
Revised 8 October 2021
Accepted 1 December 2021
Available online 29 December 2021

Keywords:
AWS dataset
Anomaly detection
Big data
Cloud computing
CSECICIDS2018

ABSTRACT

Security issues are the most critical challenges facing new technologies associated with the internet of things (IoT), big data, and cloud computing. A secure and efficient intrusion detection system (IDS) is crucial to detect security threats. Existing IDSs are known to suffer from many problems, most notably the high rate of false positive alerts, the long time required to detect attacks, and the inability to detect zero-day attacks, which can ruin companies. The weakness of IDS backend engines costs companies time in the investigation process. This paper proposes and enhances IDS detection mechanisms via two processes: using a deep neural network (DNN) model with new features for threat detection based on two assumptions related to handling zero-day attacks, with low computing power and resources, and a comprehensive solution for detection by merging the DNN model and principle component analysis (PCA) to increase security and performance. The proposed detection mechanism combines DNN, PCA, statistical, and knowledge-based approaches to offer significantly greater efficiency than existing IDS, as indicated by analytical and software results. A simulation model is used with up-to-date web attacks, distributed denial of service (DDoS), denial of service (DoS), brute force, insider infiltration, Botnet, and Heartbleed attacks. The proposed detection techniques for large networks are analyzed and complexity in the design is avoided by reducing the number of DNN model layers, thus minimizing detection time delay and false positives, while increasing security against network attacks. Integrating the proposed DNN with PCA, an innovative contribution, introduces robust IDS to significantly improve the detection time delay and security performance. The proposed model showed a 98% accuracy rate. To best of our knowledge, the highest accuracy rate stated based on a large number of attacks is 97%, which makes our model state of art.

1. Introduction

Modern ICT and internet tools have proliferated worldwide since the 1990 s and are now firmly embedded in all aspects of modern communications and the economic, agricultural, cultural, industrial, and political fields. However, legacy technology is not fit for purpose in the current technological landscape. Innovative technologies will be needed to meet the requirements of the new era in the coming years. The Internet of Things (IoT) is an example of new and innovative technology, whereby all objects may be integrated into smart systems utilizing wireless technologies, offering incredible efficiencies and convenience for users but entailing great risks in the event of malicious attacks. Due to the rapid spread of IoT technology, the expected number of devices in use will exceed 25 billion by the end of 2020 [1]. Internet users (i.e., the majority of people) increasingly produce, store, and use vast amounts of data and information, which is commensurately a highly valuable resource. Modern data storage and analysis techniques have enabled wholly new dimensions of intersection

* Corresponding author.
E-mail addresses: fawareh@yu.edu.jo (M. Al-Fawa'reh), m.alfayoumi@psut.edu.jo (M. Al-Fayoumi), shadi_nashwan@ju.edu.sa (S. Nashwan), s.fraihat@psut.edu.jo (S. Fraihat).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

between ICT fields and a vast array of areas of life not associated with the legacy internet landscape of the 20th century.

The major challenge facing computer science at this juncture is to protect and secure data from risks, as well as to protect it (and its genuine users) from malicious attacks. In-depth defense, such as multifactor authentication and the building of secure systems, does achieve protection, but it cannot detect whether a company is experiencing a zero-day attack or not. Furthermore, as well as avoiding attacks, systems must protect and gather data as evidence, as well as generally employing existing solutions such as firewalls, encryption, and intrusion detection systems. However, there are two essential problems with the existing solutions:

- They cannot detect and deal with new attacks (such as zero-day attacks);
- They generate a high rate of false positive alerts, thus increasing the time and economic costs of checks needed to verify the validity of such alerts. Consequently, researchers have proposed using cyber threat intelligence to detect stealth-based attacks. Today, threat intelligence is considered a vital asset for many organizations because it helps protect information and guarantees privacy for users. For these reasons, many restrictive approaches have been suggested. However, most of them rely on detecting adversary behaviors and then taking retroactive action post facto. One of the best solutions to detect adversary behavior is using IDS, which can potentially identify attacks before they violate user data privacy and wreak damage on user systems.

In computer security, hacking or intrusion is characterized as an arrangement of activities that seek to compromise security aspects such as the confidentiality, availability, or integrity of data in any system. Intrusion detection (ID) is a systematic process of inspecting and monitoring data in order to detect any violation, and generating alerts to protect threatened data [2].

Typically, an IDS is a collection of software or hardware resources, or both, that performs ID screening [3,4]. IDSs are divided based on the data source and the type of the detection mechanism. Depending on the position of the IDS and the source location of the data, the IDS can be categorized into many classes.

This paper focuses on the detection mechanism for the IDS. The detection mechanism is another factor used to divide IDS into three main categories: misuse detection (also known as signature), which depends on pre-existing knowledge such as byte sequences; behavior-based, which monitors the system calls or the behavior in the network [5,6,7], and the hybrid method.

2. Related work

The main application of machine learning is the detection of network anomalies—the discovery of irregular features in the data to find anomalies over time [8,9,10,11]. The effectiveness of this technique is determined by its ability to distinguish between normal and abnormal network settings. IDS technologies based on machine learning can identify existing, new, and light attacks without the need for extensive training or human interaction. They are defined as a set of techniques for recognizing data patterns and predicting future trends [12,13].

Gedam and Shikalpure [14] proposed a model for achieving high accuracy and detection rate with a low positive rate in anomaly detection using support vector machines (SVM). However, they did not actually compare their model with other models, and the work was limited to a comparison of the model with different datasets (KDD cup 99, NSL-KDD dataset and Kyoto 2006).

Jalil et al. [15] compared the efficiency of different AI approaches (such as neural networks (NNs), SVM, and DT-J48 algorithms) based on different metrics (detection rate, false alarm rate and accuracy) using a well-known dataset called a KDD-cup dataset. The experiments concluded that the DT-J48 algorithm outperformed the neural network and support vector machine (SVM) algorithms.

Masduki et al. [16] presented a new approach for detecting attacks using SVM via the classification method using a combination of 28 features, including 8 features from the dataset in [17] and 24 features from the dataset in [18] (with 4 mutual features between these two sources). Through serial experiments, the authors proved the importance of payload in detecting remote to local (R2L) attacks, and their experiments showed that SVM with 28 features achieved 96.08% accuracy. However, this work did not cover four key classes of attack types: DoS, Probe, U2R, and R2L.

Sarvari and Keikha [19] focused on a combination of ML approaches (tree, 1NN, 2NN, 3NN, and SVM) to detect abnormal behavior. The proposed combination achieves better accuracy than other approaches such as MSSGBML [20], ESCIDS [21], multi classifier [22], and PNrule [23]. They demonstrated that increasing the number of classes improved the system's accuracy. They have also demonstrated that the expansion of classifiers has a threshold and that the system's accuracy remains constant if the number of classifiers exceeds that limit. However, the authors used a small amount of data for testing their model.

Mehmood and Rais [24] used different supervised algorithms (Naïve Bayes, J.48, and SVM) to detect illegal activity in the network using a KDD-dataset based on several metrics (positive rate, precision, and false positive rate). The experiments showed that there is no single algorithm that meets the high positive rate for all classes (Normal, R2L, DoS), but the authors proved that the J.48 DT produced higher accuracy and more false positives than other approaches (Naïve Bayes, SVM). However, that work did not consider feature selection. We also consider detection approaches for both feature selection and all features.

Shukla [25] proposed three new ML approaches to uncover wormhole attacks in IOT environments using unsupervised learning (K-means clustering), supervised IDS (DT), and a hybrid approach (mix K-means and DT). The experiments show that the hybrid method achieves more accurate results than the other approaches, but the K-means algorithm works for only numerical data.

Hamid et.al [26] examined several approaches (rule-based, base rule, functions, lazy learners, tree, *meta*-algorithm, and input mapped classifier) using Weka tools and the KDD Cup 1999 dataset. The comparative analysis was based on several metrics (accuracy, recall, precision, F-measure, TP rate, TN rate, ROC area, kappa statistic, and mean absolute error). The authors used this approach both fully and with the reduction of features, proving that classification methods do not need all features, but the main limitation in this research is the limited amount of data.

Haripriya and Jabbar [27] examined various machine learning algorithms (supervised and unsupervised, reinforcement learning, etc.). The authors used method, dataset, and metrics to compare between ML approaches, but they did not study other approaches, such as fuzzy logic, ant-colony, and deep learning.

Tamimmirza [28] presented IDS with VGG-19 to detect abnormal activity based on the ISCX dataset, and the model achieved high accuracy. The main limitation of this model was its taking a feature and converting it into a black and white image.

Kanimozhi and Jacob [29] proposed a robust system to detect malicious traffic using ANN with hyper parameter optimization, achieving 99.97% accuracy. The main limitation of this research was its focus on botnet attacks, without studying the network behavior of other attacks. The second limitation was that it did not mention the number of features that were input into the model, nor did it determine the continuous or categorical data. Furthermore, the optimization phase did not give details on parameters such as the optimization function, and the preprocessing was performed via black box. Furthermore, the authors did not say whether the data were normalized or not, how they dealt with missing and categorical values, or anything about whether the data were balanced or not. Additionally, the model suffers from overfitting.

Zhou and Pezaros [30] evaluated different classifiers, such as random forest, Gaussian naive, quadratic discriminant analysis, and K-nearest neighbors. The main limitation of this research was that the data reprocessing was not explicit, and they dropped digits after the decimal point. The second limitation was that they replaced the missing values with suitable numbers, but they did not state the methods. Additionally, they did not scale the values of features, which made the comparison between the features unfair, and they did not say anything about the categorical values or data balancing. All of these drawbacks are addressed in the model presented by our paper.

To the best of our knowledge, all studies before 2010 have used ANN to study IDS, and they used the activation function sigmoid, tanh. While this model suffers from a vanishing gradient problem, the first layers train slowly compared with others. Besides this, all researchers have studied the IDS using ANN with Relu activation and a small number of records (a maximum of 4.9 million records). All studies from 2010 to 2011 suffer from slow training and miss the local minimal point of the loss function because they use an optimization function such as gradient dissent and a single learning rate. To the best of our knowledge, all studies from 2011 to 2014 miss the global minimal point and overfit. Studies from 2014 to 2015 miss the global minimal point. In 2016, it was stated that a supervised deep learning algorithm would usually perform acceptably with approximately 5,000 labeled categorical examples, and would meet or surpass human performance if trained with over 10 million labeled examples. In our dataset, we have more than 10 million items of normal flow data. Most of the previous studies used old datasets, and most of them have depended on kdd-99 and a limited number of categories (normal, R2L, DoS, etc.), while the model studied in this paper focused on a real dataset with 14 updated attacks, such as Heartbleeding attacks.

This paper presents a hybrid method for evaluating network behavior throughout the intrusion detection process, combining supervised techniques (DNN), unfiltered measurements (PCA), and statistics measures to increase the accuracy of detection of stealth attacks. Moreover, the Canadian Institute of Cyber Security (CIC) provided the CSE-CICIDS2018 Dataset, which has only been used by a few researchers as of the time of writing [31]. This dataset is used at the flow level, and statistical measurements of the flow level features are collected. The proposed method offers a basic solution to the problem of volumetric attack mitigation in the context of attacks such as DDoS, HTTP, and floods.

3. CSE-CICIDS2018 dataset

Producing and capturing realistic network traffic is considered an important pre-processing phase for comparing distinct IDSs and validating innovative methods. In particular, anomaly detection methods require extensive and up-to-date network information that resembles a true communication situation. A suitable dataset with full and accurate labels should be produced in a realistic manner. In addition, the dataset should have a good percentage of unbiased daily normal to abnormal traffic [31].

Although various standard datasets are widely available, many of them include offensive, non-modifiable, outdated, and non-reproducible attack scenarios [32]. In an effort to solve these shortcomings and create more modern traffic patterns, the CSE-CI-UNB dataset was developed by [31]. The correctly marked CSE-CI-IDS 2018 dataset exhibits real network behavior and includes a variety of intrusion scenarios. In addition, it is distributed as a complete network set with all internal paths to evaluate the loads for pre-packet inspection.

The adapted dataset includes ten days of normal and malicious network activity. The dataset was produced from two files containing abstract images of events and activities in the network. For example, communication between the source and the receiver host can be visualized over HTTP through packets sent or received and endpoint properties. This imaging creates one profile. These profiles can provide real data for protocols such as FTP, POP3, IMAP, SMTP, SSH, and HTTP [31]. Agents or individual users can use them to inject different situations into the network. In order to revitalize network activity for premium applications by changing these profiles, personal files are also exchanged with real school datasets.

In the CSE-CIC-IDS2018 dataset, descriptions are used to create datasets that contain detailed systematic descriptions of intrusions and theoretical distribution models for applications, protocols, or lower-level network entities. The CSE-CIC-IDS2018 database contains two separate models for network behavior and scenarios—B and M profiles. B profiles use different modes of computer modeling and statistical analysis to assess object behaviors for clients; the covered properties include payload size ranges, number of requests per train, and specific payload models. The simulated protocols in the tested environment include HTTPS, IMAP, HTTP, FTP, SMTP, POP3, and SSH. The majority of traffic is HTTP and HTTPS, based on the preliminary results.

M profiles attempt to define an attack scenario unambiguously. In the simplest cases, people can read these patterns and eventually implement them. Statistics for the total dataset are listed in Table 1. This attack and normal traffic scenario was executed for only ten days, with the exception of two days of infiltration attacks. Table 1 also gives an outline of the collected information, encompassing 16 million streams. The dataset marks benign and unsafe traffic streams. As we can see from Table 1, the first day of data

Table 1
CSE-CI-UNB 2018 Dataset Description.

Day (2018)	Normal flow	Attack	Attack description
14–02	667,626	• 193,360	• FTP-Brute Force
		• 187,589	• SSH-Brute force
22–02	360,833	• 249	• Brute Force-Web
		• 79	• Brute Force-XSS
		• 34	• SQL Injection
15–02	996,077	• 41,508	• DoS using GoldenEye
		• 10,990	• DoS using Slowloris
		• 139,890	• Slow HTTP Test
23–02	1,048,213	• 362	• Brute Force-Web
		• 151	• Brute Force -XSS
		• 53	• SQL Injection
		• 362	• Brute Force-Web
16–02	446,772	• 461,912	• DDoS using Hulk
		• 576,191	• DDoS using -LOIC-HTTP
28–02	544,200	• 68,871	• Infiltration
20–2	7,372,557	• 1730	• DDoS using-LOIC-UDP
		• 686,012	• DDoS using-HOIC
		• 193,360	• FTP-Brute Force
01–03	238,037	• 93,063	• Infiltration
21–02	7,372,557	• 187,589	• SSH-Brute force
		• 41,508	• DoS using GoldenEye
02–03	762,384	• 286,191	• Bot
Number of flows	19,141,630	714,290	*

mirroring contained the maximum number of attacks, at around one million flows.

This dataset contains the accompanying types and situations of assault profiles.

3.1. Brute force

In this type of attack, the attacker tries to bypass authentication by using someone's username and password, but the attacker does not know exactly what the correct credentials are, so every single character possible in the dataset is attempted in order to gain access. There are several tools available to perform brute force and password cracking, including Metasploit, Hydra, Ncrack, Nmap NSE applications, and Medusa. However, the authors in this dataset only used two modules (FTP and SSH) as the intruder machine on the Kali Linux, and Ubuntu14.0 as the target device. They use a large dictionary containing 90 million terms for a set of passwords.

3.2. Heartbleed

This attack is relatively new, first reported in 2014 (under CVE-2014–0160). It targets applications that use OpenSSL library to perform cryptography on the data, since this allows the adversary to steal the data decrypted in the memory and may contain sensitive information (username, passwords, secret keys used in encryption, etc.). The main area of this vulnerability is the implementation phase, wherein two peers use a signal to track the connection. The tool used in this attack is Heartleech, which contains important features, such as support IPv6, Tor/Socks5n proxy, and STARTTLS.

3.3. Botnet

In this attack, a large number of infected devices called zombies are connected to each other under a CC Empire. Typically, these hosts are infected by malware. When the CC launches an order, the whole army participates in the attack; usually, these bots are used in DDoS and cryptocurrency mining and distributed processing [33].

3.4. DoS

In this type of attack, the attacker uses different techniques such as packet fragmentation to establish a half-open connection to launch the attack by sending a malicious request as a normal user, so the server will allocate some resources to them, such as memory, disk space, and processing quota. The attacker repeats the same process but coming from different sources. Eventually, the server ceases being able to handle the number of requests due to the lack of resources to process them. The main goal of these attacks is to hinder CIA (availability). In this situation, the dataset used a Slowloris Perl-based device to download the internet browser.

3.5. DDoS

This attack employs the same mechanism as DoS; the only difference is that the attacker launches a massive number of requests within a short period of time from multiple devices. This scenario was generated with open source tools (HOIC) using four different computers.

3.6. Web attacks

Many types of attacks are part of this empire, and the common denominator of web attacks is that the server cannot distinguish between the command (code) and the data due to the poor validation of the input request. Web attacks scenarios were run using the well-known Damn Vulnerable Web Application (DVWA). The dataset focused on the kinds of web attacks listed below.

3.7. SQL injection (SQLi)

Here, the attacker will send a normal query with an appended malicious query, such as the updating of a table or specific column [34]. This tends to appear as follows: "SELECT National, name FROM masters WHERE id = 401 UNION SELECT *, version limit 2, 1".

3.8. Cross-Site scripting (XSS)

This attack was described by [35] as follows: "XSS empowers aggressors to infuse customer side contents into site pages saw by different clients… the equivalent source strategy may utilize a cross-site scripting defenselessness. Cross-webpage scripting completed on sites generally represented 84% of all security vulnerabilities recorded by Symantec starting in 2007". XSS can divert the victim to a dangerous location, for instance [35]: <object type = "text/x-scriwptlet" data = "http:/dahamTeccga.com/xss.html">

3.9. Inclusion vulnerabilities

Remote File Acronyms created by RFI and Local File Inclusion (LFI) are common weaknesses in poorly written web applications [36]. These vulnerabilities arise when a user sends input to files or upload files to the server through a web application. Weaknesses in RFI are simpler but less easy to exploit. The hacker executes a code stored on their computer instead of accessing a file on the local machine. The LFI allows the hacker on the victim's computer to read (and sometimes execute) files. This can be very dangerous, because the attacker can gain access to sensitive information if the application server is misconfigured or additionally run with privileges.

Here, the attacker sends a normal request appended with a system command, such as (mog.faw@psut; cat /etc/passwd). The Network is infiltrated from the inside. This is considered one of the most dangerous types of attack as the attacker is trying to gain access to the internal network. The attacker compromises one machine using different techniques, such as social engineering, or client- or server-side attacks, etc., to then launch an attack on different networks; only one has access to the compromised system. A vulnerable application (such as Adobe Acrobat Reader 9) can be exploited under this scenario. Fig. 1 shows the number of attacks per class. In addition, we see that the dataset is unbalanced.

The network testbed architecture comprises 500 machines divided into five separate LANs for building a realistic network topology, namely: Department of R&D, Department of Management Department of Technicians, Secretary and Department of Operations, Department of IT, and database spaces. In this dataset, the authors deployed lists of different Microsoft operating systems, such as Windows 8.1 and 10, for all organizations except the IT department; all devices in the IT department are Linux OS. Various MS Windows servers, such as 2012 and 2016, were implemented for server farms [31]. Despite the various hacking scenarios in the real-life network, there are some limitations to this dataset. For example, a large portion of the IP packets generated by data management and network monitoring operations are un-named and anonymous. Additionally, when flow records are retrieved from the dataset, some streams are found to contain infinity and NAN values.

This can be explained by the characteristics of many connections. Some attacks involve a multi-stage process, such as checking the network to obtain information about the victim's machine. For

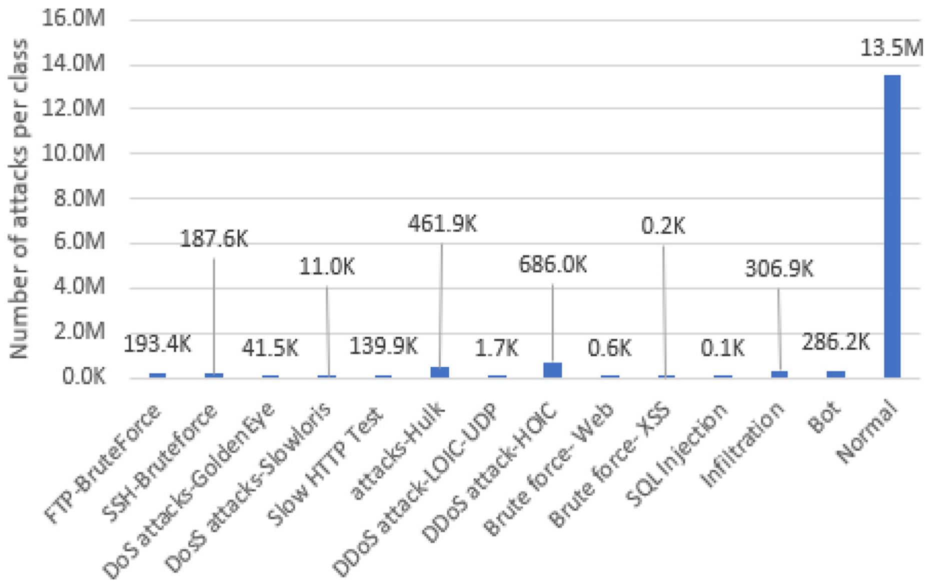

Fig. 1. IDS 2018 dataset—number of attacks by type.

example, attackers use the address resolution protocol (ARP) to obtain information and some devices in the attack network; in this case, the L2 connection using ARP remained because the requests di not use ports. Thus, the transmitter and receiver port numbers contain infinity and NAN values for the streams that contain this interaction protocol. The dataset was pre-processed and cleared to address these issues. All preliminary treatment steps are shown in the following sections. Another limitation in this dataset of web attacks is their limited number compared with the other classes.

4. Conceptual framework and methodology

This section describes the methodology of the proposed model architecture. Firstly, a summary of the workflow of the system is presented, with a brief description of the architectural models (i.e., PCA and DNN) that inspired our multi-layer architecture. The proposed architecture and its implementation are subsequently described. As previously stated, this research proposes an anomaly detection system that analyzes flow-level traffic and captures attacks. The proposed system combined four well-known techniques:

- Knowledge experience;
- Statistical models for data acquisition and the transfer of data from the packet to flow levels;
- Principle component analysis for dimensionality reduction. Deep neural networks for classification.

The workflow pipeline used in the experiments is highlighted in Fig. 2. This hybrid model comprises six steps:

4.1. Data collection

Network traffic information is often collected in the form of initial packet capture (PCAP) in network switches or routers [5,35]. PCAP information includes full TCP/IP packet data sent or received on a specific network computer. While in some cases full packet capture information may be useful, this has a high storage cost.

An alternative (or addition) to PCAP data is NetFlow, used to describe PCAP data in terms of top-level network flows. NetFlow data have the benefits of wider and smaller disk space requirements, while PCAP data require much larger but more expensive disk space [37].

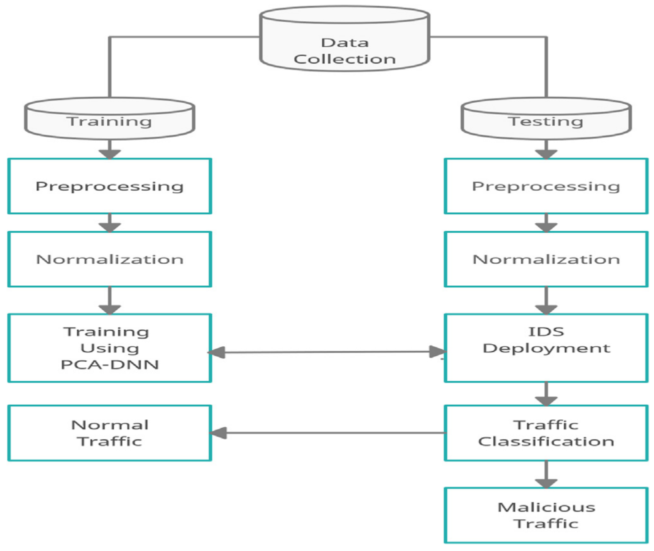

Fig. 2. Proposed model.

Typically, NetFlow logs consist of the protocol, byte number, source interface, and source destination IP and port. They can extract many fields from PCAP, depending on the software configuration used to convert PCAP to NetFlow data. NetFlow records are used as notes by the ML algorithms used in this work. Therefore, they must first be converted to NetFlow if there are any PCAP entry data.

In practice, it is best to run an algorithm that takes NetFlow as the input. Cisco has created a NetFlow protocol that is commonly used for further analyses by many network servers, routers, switches, and firewalls to export raw traffic data to the data flow [38].

The advanced IETF specification is called IPFIX, an abbreviation of IP Flow Information Exchange, which is similar to the NetFlow standard [38]. The difference is that it permits additional information that is normally provided to Syslog in addition to fields of variable length, allowing for data collection such as that pertaining to HTTP hosts, URLs and emails. More intensive traffic logs are simply referred to as "flow data" for the purposes of this work [38].

A variety of tools are available to convert raw PCAP data into data flow, such as the CICFlowMeter tool, developed by the University of New Brunswick [39]. YAF was developed by the Computer Emergency Response Task Force (CERT) [40], and is part of the Network Parking Recognition Security (NetSA) [39] toolkit for tracking large networks.

PCAP packet data are converted into two-way streams by YAF, which subsequently outputs the streams to an IPFIX-based file format. YAF converts PCAP packet data into two-way streams, and then exports the streams to the IPFIX-based file format. The authors included this dataset in the CSE-CIC-IDS2018 dataset in the form of both PCAP and flow records computed by [31]. CICFlowMeter is a raw PCAP-based network traffic flow generator. It generates two-way streams, the first of which establishes the forward (source/destination) and backward (destination/source) paths [41].

The main difference between the CIC IDS 2018 dataset and the ISCX IDS 2012 and CIC IDS 2017 datasets [42] is the number of classes and network flow features available compared to the ISCX IDS 2012 and CIC IDS 2017 datasets. The ISCX IDS 2012 dataset has 14 total features, while the datasets labeled CIC IDS 2017 and CSE CIC IDS 2018 have 81 available features each. These additional features consist primarily of comprehensive flow statistics and packet information calculations for each flow created using the CICFlowMeter open source tool.

The main difference between CIC IDS 2017 [43] and CSE CIC IDS 2018 [31] is the quantity of the data and the number of days required to launch the attacks (relating to the number of machines used). Another major difference in the CIC IDS 2018 dataset is that it is made available in label-based CSV files with pre- processed packet data in the form of flow records. For each flow file, the CICFlowMeter provides the corresponding protocol number, IP address, unique Flow ID, IP source/destination, timestamp, and tag. It also produces 77 statistical features, such as size, number of bytes, and number of packets, which are also measured separately in both the forward and reverse directions.

In this paper on resources limitation (processing and storage), we removed the data from the fourth day (20–02-2018). The Network Aggregation CIC Flow Meter is a network traffic flow generator written in Java that provides more stability when choosing the properties to calculate, introducing new properties and a stronger capacity for monitoring the length of the stream timeout. Biflow flows are produced, with the first beam setting the directions forward (source to destination), and vice versa. In total, 83 statistical features, such as source IP, source port number, number of packets, duration, etc., were calculated individually in the forward direction and in the reverse [44]. The CICFlowMeter-V3 is able to obtain more than 80 features, as mentioned in Table 2. Additional information about the dataset is presented in [31].

4.2. Data preprocessing phase

Once the data are in the form of flow data, the further cleaning, conversion, and preparation of the data for use as input into the deep learning algorithm are performed. This step includes tasks such as making sure that the data do not contain invalid characters, removing fields with empty values, removing or changing values (not the number), and removing duplicate columns. The main reason for preprocessing is that the data are in different formats and are collected from different locations; it also ensures the accuracy and effectiveness of the model being trained on this data.

A common practice during this stage is to normalize or extend the range of continuous values between all properties, so the deep learning algorithm trains data in the same job space. Two basic forms of normalization are widely used: standardization (or Z degree normalization) and min–max expansion. Standardization results in measurable properties with standard normal distribution values. Standard scores for features (also known as z -score) are calculated as in Equation (1), where U is the mean and S is the standard deviation from the mean:

$$\boldsymbol{Z} = (\boldsymbol{X} - \boldsymbol{U})/\boldsymbol{S} \tag{1}$$

Table 2
Features Description.

#	Feature Name	Description	Feature Type
F1-F2	Flow ID,Flow_dur	Flow record identifier, flow duration	continuous
F3-F6	(Src/Dst) IP, (Src/Dst) Pt	(Source/destination) IP address, (source/destination) port #	categorical
F7-F8	Pt/ TS	Protocol, timestamps	categorical
F9-F10	Tot_(fw/bw) _pk	Total # of (forward/backward) packets	continuous
F11-F12	Tot_(l)_(fw/Bw_pkt)	Total lengths of (forward/backward) packets	continuous
F13-F22	Tot_(l)_(fw/Bw_pkt) (max/min/avg/std/mean)	(Max/min/avg./standard deviation/mean) Packet size in (Forward/ backward) direction	continuous
F23-F26	Fl_iat_avg/std/max/min	Average/standard deviation/maxi-mum/minimum time between two flows	Continuous
F27-F36	(Fw/Bw) _iat_ (tot/avg/std/max)	(Total/mean /maximum /minimum /standard deviation) time between two packets that are sent in (Forward/Backward) direction	Continuous
F37-F40	(Fw/bw) _ (psh/urg) _flag	# of times that (URG/Push) flags are set to 1 in (forward/backward) direction	Continuous
F41-F48	(Psh/ftn/syn/rst/ack/cwe/ece/urg)_cnt	#ofpacketswith (PUSH/FIN/SYN/RST/ACK/CWE/ECE/Urg) flags	Continuous
F49-F52	Pkt_len_ (min /max/avg/std)	(Minimum/maximum/average/standard deviation) flow length	Continuous
F53-F54	(Fw/bw) _pkt_s	# of packets in (forward/backward) per sec	Continuous
F55-F58	(Fw/bw) _(byt/pkt) _blk_avg	The Average # of (packets/bytes) bulk rate in (forward/backward) direction	Continuous
F59-F62	Subfl_(fw/bw) _(pk/byt)	The average # of packets/bytes for sub flow in forward/backward direction	Continuous
F63-F67	Atv_(min/max/std/avg)	(Minimum/maximum/standarddeviation/ average) time when a flow was in an active mode before switching to the idle state	Continuous
F68-F71	Idl_(min/max/std/avg)	(Minimum/maximum/standarddeviation/ average) time when a flow was in idle mode before switching to the active mode	Continuous
F72	Down up ratio	Upload and download ratio	Continuous
F73	Pkt_size_avg	Packets size average	Continuous
F74-F77	(Fw/bw) _seg_(avg/min)	The (average/minimum) packet size in (forward/backward) direction	Continuous
F78-F79	Fl_(byt/pkt) _s	# of (packets /bytes) transferring per s	Continuous
F80-81	(Fw/bw) hdr_len	Totalbytesintheheaderin (forward/backward) direction	Continuous

The min–max scale, known as standardization, consists of scaling information to a fixed range, usually [0.1]. The only problem with this type of measurement approach is that the smaller standard deviations will reduce the effectiveness of the extreme values. Equation (2) is used to implement the min–max measurement:

$$X_{min} = (X - X_{min})/(X_{max} - X_{min}) \tag{2}$$

In this research, data standardization is carried out by subtracting the mean from the dataset and dividing the remaining dataset by the square roots of individual values. As explained earlier in section 4, the number of potential joint values across all of these categorical outputs is 14. Therefore, in order to experiment with these super dimensional labels, they must be converted to floating point values for use in the neural network. To do this, there are a number of basic and more complex approaches available. The following techniques have been described to work with these highly significant categorical variables: one hot coding OHE, one hot segmentation trick, and entity inclusion. OHE is a common technique for converting categorical output variables into continuous signs. This system works by creating a new variant of the original categorical mark for every possible unique value. There are two main problems with using OHE in neural networks [41]: OHE ignores links between categorical properties and treats them distinctly from one another. In our case, though, this will not affect coding (i.e., it only affects the label and not the features). The OHE of slow cardinal properties can be slow, and requires a great deal of computational power to complete. Because we have only 14 categories, computation will not take much time.

4.3. Data splitting

The dataset is divided with 20% of the data placed in a test dataset and 80% in a training dataset. Since this dataset is highly unbalanced, as shown in Fig. 3, it is important to ensure that there is the same percentage of malicious flows in the training and test datasets, respectively. To achieve this, the dataset is divided in such a way as to ensure that the distribution of benign and malicious traffic in both the training and test datasets is balanced. Cross-validation is used during training repetition with the K-fold equal to five [45].

4.4. Principal component analysis (PCA)

The well-known technique of PCA is a mathematical procedure that uses orthogonal transformation to convert a set of observations of possible associated variables into a set of non-linearly related variable values, called principal components. PCA is applied primarily in dimensionality reduction. It is a non-supervised projection method that can translate from a complex larger area to a smaller new area with minimal information loss [46]. If the original data matrix is defined as $X_d * N$, then PCA creates a $W_d * k$ projection matrix representing the change in the base, where d represents the original dimensional vector of data x_i , k is the dimension of the reconfigured data vectors $y_i = W^T x_i$ after the dimension reduction, and N is the x_j data vector number.

In the proposed approach, PCA was selected over alternative approaches such as t-SNE for the following reasons:

- Stochasticity of final solution. PCA is deterministic; t-SNE is not;
- Incomplete data. Natively, t-SNE does not deal with incomplete data. In all fairness, PCA does not deal with them either, but numerous extensions of PCA for incomplete data (e.g., probabilistic PCA) are available, and are included in almost all standard modeling routines. t-SNE currently cannot handle incomplete data.

Detection performance can be enhanced by reducing the space of features and noise characteristics. PCA is one of the more widely used methodologies for selecting the main features from the dataset and eliminating dispensable ones. Different numbers of key components have been evaluated in this paper, ranging from 2 to 78, with the optimal results derived when n = 12; n (the number of features) includes F2, F4, F9, F10, F11, F12, F13, F14, F15, F17, F18 and F19, as shown in Table 2.

From a security point of view, the above statistical feature can detect any change towards abnormal behavior, such as would be seen in a cross-fire attack.

4.5. Deep neural networks (DNNs)

DNNs are amongst the most commonly used machine learning approaches. They can reverse increasingly complex functions by merging more layers and more units per layer into a neural network [41]. A DNN can be used to identify normal and malicious traffic patterns hidden within large quantities of structured data.

Deep neural networks are an effective tool when conducting supervised learning for network intrusion detection tasks, especially in the context of the continued growth in computing power.

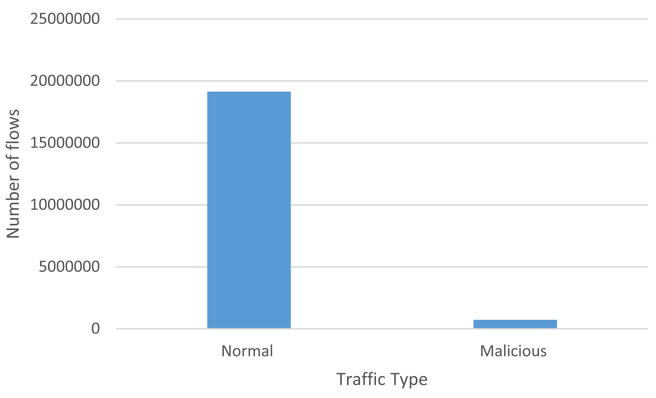

Fig. 3. Network traffic distribution.

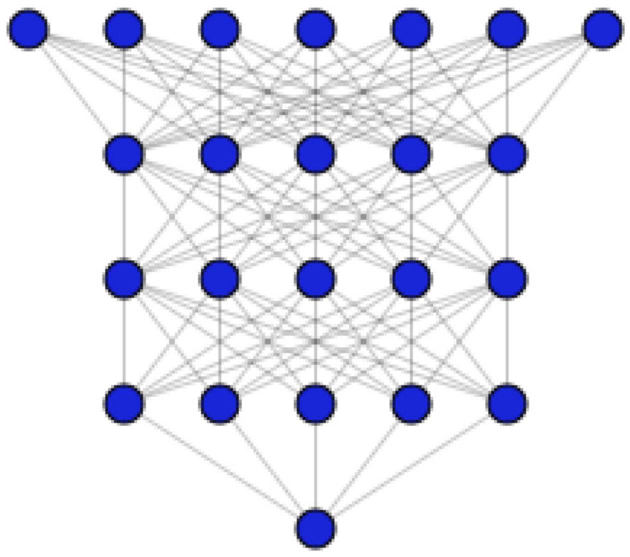

Fig. 4. DNN architecture.

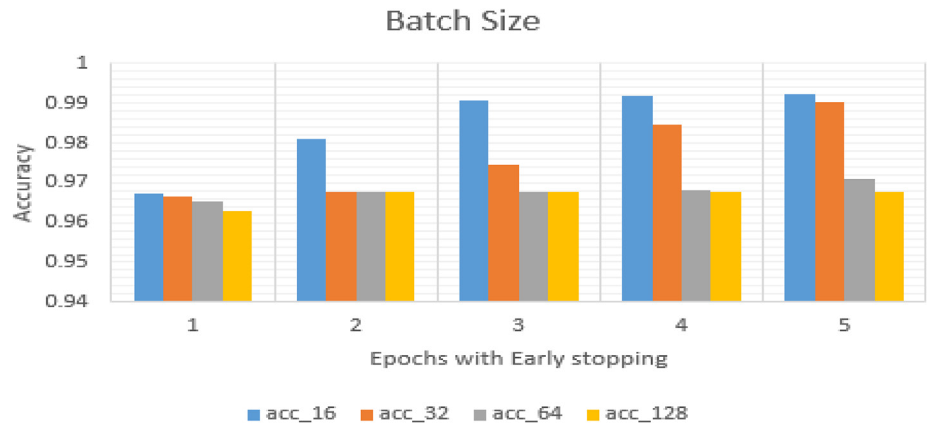

Fig. 5. Effect of the batch size on the loss function.

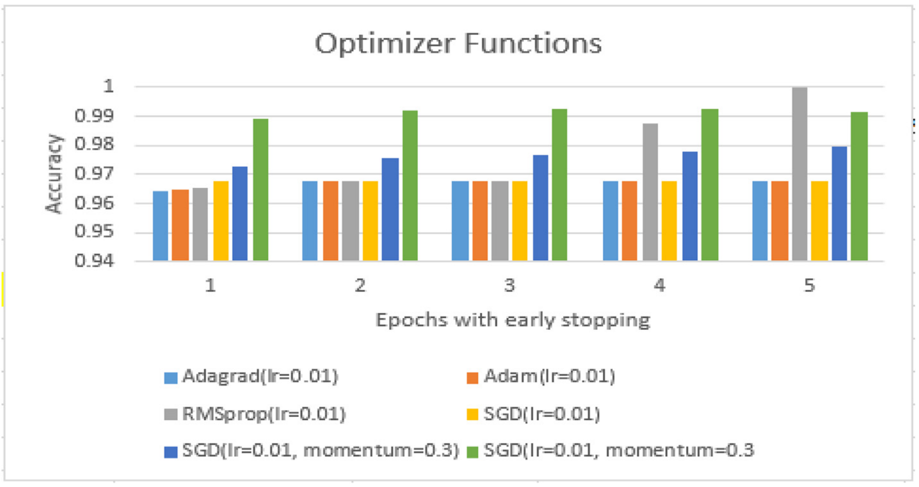

Fig. 6. The effect of the optimizer function on the loss function.

4.5.1. Model definition

In deep learning, the hyperparameters are parameters that control the learning process. These excessive parameters are calibrated based on the validation set derived from system output feedback, and include batch size, epoch, loss functions, activation functions, data normalization techniques, number of units per layer, number of hidden layers, learning rate, regularization technique (L1, L2), and optimization algorithms. The values of those hyperparameters are obtained after running many experiments with different values, as shown in Fig. 5. It can be seen that the loss function is minimal when the model uses the mini batch SGD = 16.

In a DNN, an activation function is a function that determines the output value of each neuron. It is extremely important to the model's overall performance. Different activation functions, such as Sigmoid, ReLU, Tan, SoftMax, and Elu, have been used.

As an activation function, the rectified linear unit (ReLU) defined by Eq. (3) calculates the sum of all weighted inputs after setting the negative values to zero and outputs a sum in the range of $(0, +\infty)$.

If the input is positive, the ReLU activation function outputs it directly; otherwise, it outputs zero. This model is widely used in various types of neural networks since it is simple to train and attains good performance [47].

$$ReLU(x) = \begin{cases} xifx > 0 \\ 0.01xotherwise \end{cases} \tag{3}$$

Eq. (4) defines the sigmoid function as a particular case of the logistic function defined. It has a positive derivative at each location and is bounded. The sigmoid activation function is a popular function for neural networks due to its non-linearity and the computational simplicity of its derivation [48]. The input to the Sigmoid function is converted into a value between 0 and 1. Inputs that are greater than 1.0 are converted to 1 and values below 0 are converted to 0.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

The Softmax activation function is largely used in the output layer in neural network models, and specifically in the classification task. It converts the weighted sum values provided by the hidden layers into probabilities that add up to one, which correspond to each class. The Softmax activation function is defined by Eq. (5) [49].

$$Softmax(V_{input}) = \frac{e^{V_{input}}}{\sum_{\#class} e^{V_{input}}} \tag{5}$$

Fig. 6 shows that the loss is minimal when using the RMSProp optimizer. Fig. 7 shows that the accuracy of the model is the highest when the learning rate = 0.01.

After extensive experiments, the DNN consisted of three hidden layers with 64 units per layer, as shown in Fig. 4. The activation function in each hidden layer is the rectified linear unit (ReLU), with a Softmax activation function for the output layer in multi-classification and Sigmod foinr binary classification. After the testing of a set of optimizer functions, such as Adam and SGD, edged with definite momentum, RMSProp was found to perform better, with better results in both the training and testing phases.

On the other hand, the mean squared error (MSE) was used as a loss function for the optimization algorithms in order to minimize the error between predictions and expected values, and was computed by Eq. (6). The MSE metric is one of the most popular metrics for evaluating the performance of machine learning by calculating the difference between the predicted and expected values [50].

$$MSE = \frac{1}{n}\sum_{i}(Predicted_i - Expected_i)^2 \tag{6}$$

5. Experiments and models evaluations

In this section, a DNN technique with and without dimensional reduction (PCA) applied to the CSECICIDS 2018 dataset will be briefly summarized. The experimentations were conducted on a 64-bit Windows 10 PC with 16 GB RAM and 2.60 GHz CPU, and deep learning was performed using Python 3.7.3 and Numpy 1.16.2.

5.1. Assessment method and metrics

IDS assessment is a critical task as systems must demonstrate how well they compare to other IDS tools [51]. Measurements of intra-network anomalies, accuracy, confusion matrices, sensitivity,

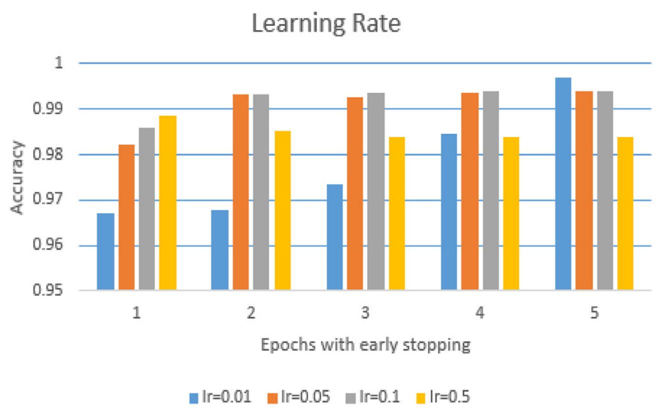

Fig. 7. Learning rate.

specificity, ROC curves, and F1 are widely used as evaluation measures. In addition, the model's time and location complexity are useful when assessing the efficiency of real-time IDs. More metrics are used, such as the accuracy of false positives (FP), true negatives (TN), true positives (TP), and false negatives (FN). The confusion matrix is another way to compare the actual category tag with the expected designations and display the values of TP, TN, FP, and FN. A number of metrics are identified in this paper to test the effectiveness of deep learning methods [52]. First, the basic terms are outlined below:

TP—represents the number of observations that are reliably expected to be positive (for example, the ground truth is "harmful", and the expectation is thus "harmful" as well);

TN—represents the number of observations that are accurately predicted to be negative (for example, the ground reality is "benign" and the prediction is "benign" as well);

FP—represents the number of incorrect observations predicted to be positive while in reality they are negative (for example, the basic truth is "benign", while there is a "harmful" expectation);

FN—the number of incorrect samples that are predicted to be negative (for example, the ground truth is "malicious" but is expected to be "benign").

The confusion matrix can be used to represent the output of a supervised learning classification algorithm. Predicted observation metrics can be defined as follows (Eq. (7, 8, 9, 10)) in the context of network intrusion detection [51].

$$TruePositiveRate(TPR) = \frac{TP}{TP + TN} \tag{7}$$

$$TrueNegativeRate(TNR) = \frac{TN}{TN + FP} \tag{8}$$

$$FalseNegativeRate(FNR) = \frac{FN}{FN + TP} \tag{9}$$

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \tag{10}$$

The accuracy is the ratio of the total number of valid TP + TN predictions to all the TP + FP + FN + TN predictions.

Evaluation of Training Time: The tests are performed using the time required to train the computer system models. This is achieved by using the Python process time command to define the start and end times, then subtracting the end time from the start time.

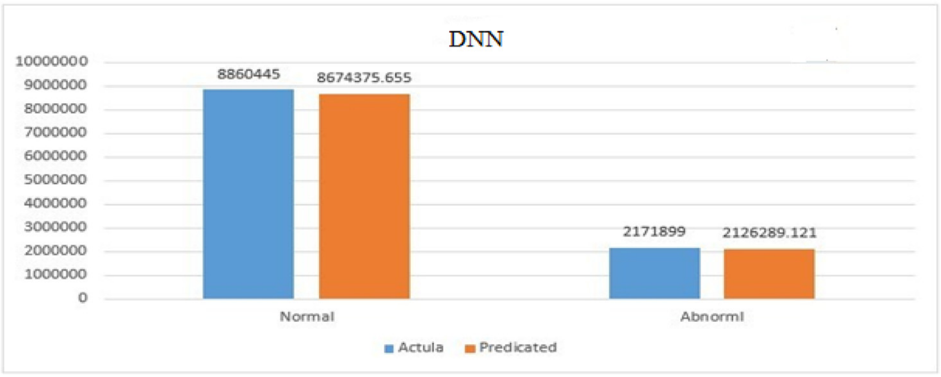

Fig. 8. DNN without PCA.

5.2. Experiments

All tests were performed, and the results can be considered satisfactory. There is always room for improvement, and it is possible to use different configurations of parameters and tests relative to the power of the GPU processor, etc. The experimental results show that all algorithms were of good accuracy, and the parameters were improved by testing with ranges of values before setting the best values. Different parameters were evaluated when performing experiments, and the strongest were used as model parameters. The experiments are divided into four phases:

- Binary classification with PCA
- Binary classification without PCA
- Multi-classification with PCA
- Multi-classification without PCA

Table 3
Accuracy of DNN algorithm in percentages.

Correct classification	Incorrect classifications	Time Consumed (in sec)
97.97%	2.23%	782

5.2.1. Binary classification

Theoretically, binary classification should run faster compared to multi-layer categorization. Firstly, the DNN algorithm was used in experiments and achieved a good prediction accuracy based on the hyperparameter optimization model. In Fig. 8, the actual and predicted numbers of normal and attack flows are represented, and the result indicates that actual attacks and predicted attacks behave similarly. The evaluation achieves high accuracy in both normal and anomalous cases, with less than a thousand error ratings being identified. The calculation achieves an overall accuracy of 97.97% as regards the percentage of discovered numbers. This means that, if the accuracy is 97.97%, the error rate is 2.23%, which (while not perfect) is a very similar and good result. The performance of the algorithm is illustrated in Table 3.

The overall accuracy of detection was 97.97%—the highest result among all tested algorithms. The performance is presented in Table 3.

As Fig. 9 shows, the overfitting points in the fourth epoch and the early stopping function are employed here to avoid overfitting.

The second part of the DNN experiments involved using PCA with n = 2 (where n is the number of the dimensions), which

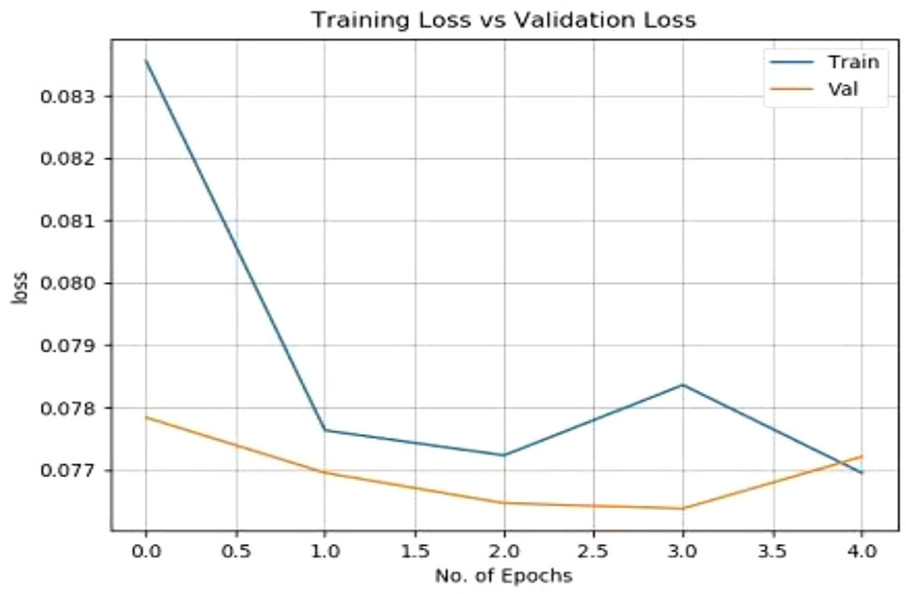

Fig. 9. Early stopping point.

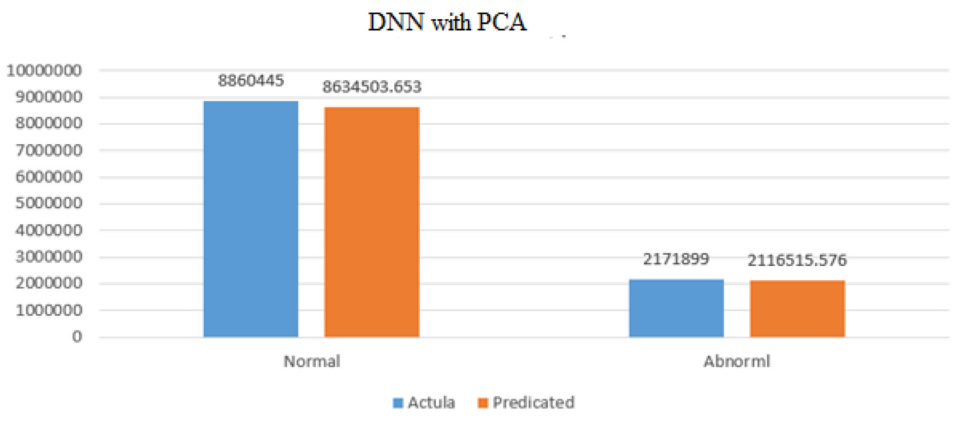

Fig. 10. DNN with PCA.

achieved a high accuracy. As can be seen from Fig. 10, the numbers of actual attacks and predictions were close.

The overall accuracy when applying PCA was 97.49%, reducing the training time by half, as seen in the last experiment. The performance of the algorithm when using PCA is presented in Table 4. Figs. 11 and 12 show the related confusion matrix table.

As we can see in Table 5, the training time was reduced by half when we used PCA (n = 2), and the accuracy was also reduced to 96.76%.

5.2.2. Multinomial classification

The multinomial classification results are shown in Table 6. It can be seen that the greatest accuracy achieved without PCA was 98%, but this requires a lot of time in the processing phase, while the best performance with PCA was 98% when n = 12. Time consumption was calculated in this experiment, as this is a fundamentally important factor. This involves checking how fast the different codes and algorithms are running using a timestamp provided by the training computer. It was expected that binary results would take less time to generate than multiclass results because of the lower number of classes available for identifying the samples.

Table 4
Training time with and without PCA.

Experiment	Time Consumed (in sec)	Training Accuracy	Testing Accuracy
Binary classification without PCA	781.9	97.49	97.49%
Binary classification with PCA	472.63	96.82	96.76%

Table 5
Accuracy of DNN with PCA algorithm in percentages.

Correct classifications	Incorrect classifications	Time Consumed (in sec)
96.79%	3.21%	473

Table 6
DNN with several PCA dimensions (multi-classification)

Experiment	Time Consumed (in s)	Training Accuracy	Testing Accuracy
DNN without PCA	1951.30	97.96	98% (+/−0.00)
DNN with PCA (n = 2)	1302.15	95.01	95% (+/−0.01)
DNN with PCA (n = 4)	1521.68	96.97	97% (+/−0.01)
DNN with PCA (n = 8)	1496.32	97	97% (+/−0.00)
DNN with PCA (n = 12)	1458.78	98	98% (+/−0.00)

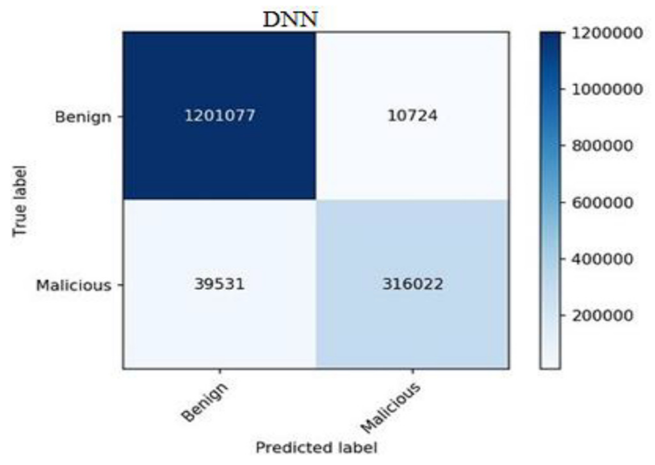

Fig. 11. DNN with PCA.

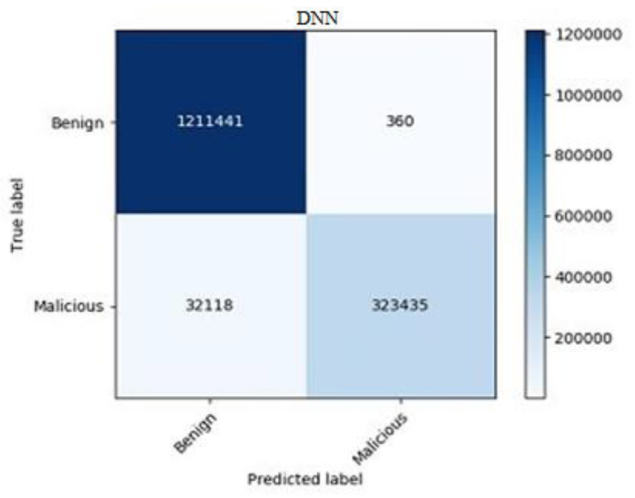

Fig. 12. DNN without PCA.

5.3. Discussion and analysis

When using flow-level features extracted from network traffic without PCA reduction approaches, and with the removal of important features such as source/destination IP address and a reliance on statistical characteristics, the DNN performs well in classifying malicious and benign flows (achieving a 98% accuracy rate). This compares to the experiment including PCA, and specifically the features F2, F4, F9, F10, F11, F12, F13, F14, F15, F17, F18 and F19, which achieved 98% as well. Therefore, training a DNN by omitting features allows it to identify the source, destination, traffic type, etc., and give it statistical and network features. It will save costs in the training phase; hence it may be usable in a pre-trained model, of a kind not seen before.

Having evaluated the performance of the proposed models with and without dimensionality reduction, and compared the results with [29,30,53] in the detection of malicious activity, the techniques were examined based on the accuracy of all attacks, as shown in Table 7. According to the experiments conducted, our hybrid model outperforms the current state of the art techniques [32].

As shown in Table 7, the models in [29,53] used only one attack (Botnet) to train their models, while this paper used 20 attacks. As such, the proposed models can handle different attack surfaces. Therefore, the main advantage of the proposed model over those in [30,32] is the ability to detect different types of attacks with a high accuracy, rate while other models can only detect botnet attacks very well. Furthermore, the proposed model reduced the consuming time during the training process, as shown in Table 7.

6. Conclusions

This paper addressed the challenge of detecting attacks in a huge network. We have implemented, designed, and developed a hybrid model for anomaly detection in large networks, particularly in cloud environments. Because of the volume of data involved in modern networks, high-speed processing at the packet level can be very expensive. Our system analyzes network traffic at the flow level (rather than single-packet information) to cope with this issue. The proposed model has been validated by detecting real

Table 7
Summary of classification performance of our model and the state of the art models.

Model	Method	Attack Type	Testing Accuracy	Time Consumed (s)
[29]	ANN	Bot	99.97%	186.4
[53]	KNN, SVM, DT, RF, NB	Bot	99.73%, 99.98%, 99.9%, 99.83%, 99.92%	189.9, 18050, 9.84, 17.77, 1.69
[30]	RF	All attacks in [31]	96%	14,200
[32]	RF, KNN, DT	All attacks in [31]	97%	1436.51, 259200, 1518.72
Our model	DNN	All attacks in [31]	98%	1951.3
Our model	PCA-DNN	All attacks in [31]	98%	1458.78

anomalies in real traffic traces. It relies mainly on DNN and PCA. DNN can learn features on its own, deriving data for training, and neural model networks can self-extract features without relying on other technologies (such as manual methods) to obtain engineering features. PCA is helpful in the processing phase. Manual feature engineering is still important, enabling the use of fewer resources to solve the problem in a more elegant way. Using an advanced cloud- based dataset for intrusion detection is efficacious in applying fully connected feedforward and backward DNN for malicious and benign detection, and the flow level was explored using both supervised and non-supervised techniques. Our experiments demonstrate that hybrid models can be used to detect anomalies when educated on benign flows, allowing for the detection of malicious flows with low false positive frequency.

References

[1] Gartner, "Gartner Identifies Top 10 Strategic IoT Technologies and Trends," 2010. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends. [Accessed: 14-Dec-2019].
[2] S. Al-Saqqa, M. Al-Fayoumi, and M. Qasaimeh, "Intrusion Detection System for Malicious Traffic Using Evolutionary Search Algorithm," *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, vol. 14, no. 5, pp. 1381–1389, 2021.
[3] S. S. Rajan and V. K. Cherukuri, "An overview of intrusion detection systems," *Retrieved May*, vol. 12, no. 3, pp. 559–563, 2010.
[4] Liao Hung-Jen, Richard Lin Chun-Hung, Lin Ying-Chih, Tung Kuang-Yuan. Intrusion detection system: a comprehensive review. J Network Comput Appl 2013;36(1):16–24.
[5] Corsini J. Analysis and evaluation of network intrusion detection methods to uncover data theft. UK: Edinburgh Napier University; 2009.
[6] Al-Fayoumi Mustafa, Alwidian Jaber, Abusaif Mohammad. Intelligent association classification technique for phishing website detection. Int Arab J Inform Technol 2020;17(4):488–96.
[7] Kobbaey Thaeer, Hamzaoui Raouf, Ahmad Shakeel, Al-Fayoumi Mustafa, Thomos Nikolaos. Enhanced collision resolution and throughput analysis for the 802.11 distributed coordination function. Int J Commun Syst 2021;34(16). doi: https://doi.org/10.1002/dac.v34.1610.1002/dac.4953.
[8] Denning DE. An Intrusion-Detection Model; 1987.
[9] Y. A. Yaseen, M. Qasaimeh, R. S. Al-Qassas, and M. Al-Fayoumi, "Email Fraud Attack Detection Using Hybrid Machine Learning Approach," *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, vol. 14, no. 5, pp. 1370–1380, 2021.
[10] Alawneh E, Al-Fawa'reh M, Jafar MT, Al Fayoumi M. Sentiment analysis-based sexual harassment detection using machine learning techniques. In: 2021 International Symposium on Electronics and Smart Devices (ISESD). p. 1–6.
[11] Karaymeh A, Ababneh M, Qasaimeh M, Al-Fayoumi M. Enhancing data protection provided by VPN connections over open WiFi networks. In: 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS). p. 1–6.
[12] Dietterich T, Bishop C, Heckerman D, Jordan M, Kearns M. Introduction to machine learning second edition adaptive computation and machine learning. second ed. MA: MIT Press; 2010.
[13] Modi Chirag, Patel Dhiren, Borisaniya Bhavesh, Patel Hiren, Patel Avi, Rajarajan Muttukrishnan. A survey of intrusion detection techniques in Cloud. J Network Comput Appl 2013;36(1):42–57.
[14] A.G. Gedam, S.G. Shikalpure, Direct kernel method for machine learning with support vector machine, *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies, ICICICT 2017*, vol. 2018-Janua, pp. 1772–1775, 2018.
[15] Jalil KA, Kamarudin MH, Masrek MN. Comparison of machine learning algorithms performance in detecting network intrusion. In: ICNIT 2010–2010 International Conference on Networking and Information Technology. p. 221–6.
[16] B. W. Masduki, K. Ramli, F. A. Saputra, and D. Sugiarto, "Study on implementation of machine learning methods combination for improving attacks detection accuracy on Intrusion Detection System (IDS)," *14th International Conference on QiR (Quality in Research), QiR 2015 – In conjunction with 4th Asian Symposium on Material Processing, ASMP 2015 and International Conference in Saving Energy in Refrigeration and Air Conditioning, ICSERA 2015*, pp. 56–64, 2016.
[17] S. Mukkamala, G. Janoski, A. Sung, Intrusion detection using neural networks and support vector machines, in: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, 2002, vol. 2, pp. 1702–1707 vol.2.
[18] Natesan P. Multi stage filter using enhanced adaboost for network intrusion detection. Int J Network Secur Its Appl 2012;4(3):121–35.
[19] H. Sarvari and M. M. Keikha, "Improving the accuracy of intrusion detection systems by using the combination of machine learning approaches," in *Proceedings of the 2010 International Conference of Soft Computing and Pattern Recognition, SoCPaR 2010*, 2010, pp. 334–337.
[20] Al-sharafat WS, Naoum R. Development of genetic-based machine learning for network intrusion detection. Int J Comput Inform Eng 2009;3(9):20–4.
[21] Toosi AN, Kahani M. A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. Comput Commun 2007;30(10):2201–12.
[22] M. Sabhnani, G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context," in *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications*, 2003, pp. 209–215
[23] R. Agarwal and M. V. Joshi, "PNrule: A New Framework for Learning Classifier Models in Data Mining (A Case-Study in Network Intrusion Detection)," 2013.
[24] Mehmood T, Rais HBM. Machine Learning Allgorriitthms IIn Conttextt Off Inttrusiion Dettecttiion. In: Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on. p. 369–73.
[25] P. Shukla, "ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things," *2017 Intelligent Systems Conference, IntelliSys 2017*, vol. 2018-Janua, no. September, pp. 234–240, 2018.
[26] Y. Hamid, M. Sugumaran, L. Journaux, "Machine learning techniques for intrusion detection: A comparative analysis," *ACM International Conference Proceeding Series*, vol. 25-26-Augu, no. December 2017, 2016.
[27] L. Haripriya and M. A. Jabbar, "Role of Machine Learning in Intrusion Detection System: Review," in *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, 2018, no. Iceca, pp. 925–929.
[28] Tamimmirza, "GitHub – tamimmirza/Intrusion-Detection-System-using-Deep-Learning: VGG-19 deep learning model trained using ISCX 2012 IDS Dataset," 2018. [Online]. Available: https://github.com/tamimmirza/Intrusion-Detection-System-using-Deep-Learning. [Accessed: 17-May-2021].
[29] Kanimozhi V, Jacob TP. Calibration of various optimized machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. Int J Eng Appl Sci Technol 2019;4(6):2143–455.
[30] Q. Zhou and D. Pezaros, "Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection – An Analysis on CIC-AWS-2018 dataset," 2019. [Online]. Available: http://arxiv.org/abs/1905.03685.
[31] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, 2018, no. Cic, pp. 108–116.
[32] Al-Fawa'reh M, Al-Fayoumi M. Detecting stealth-based attacks in large campus networks. Int J Adv Trends Comput Sci Eng 2020;9:4262–77.
[33] Faek R, Al-Fawa'reh M, Al-Fayoumi M. Exposing bot attacks using machine learning and flow level analysis. In: International Conference on Data Science, E-learning and Information Systems. p. 99–106.
[34] Mohd Yunus MA, Brohan M, Mohd Nawi N, Salwana E, Najib N, Liang C. Review of SQL injection: problems and prevention. JOIV 2018;2:215.
[35] V. Xiii, "Symantec Internet Security Threat Report trends for July-December 07 Executive Summary," 2008.
[36] Cwe.mitre, "CWE – CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion') (3.4.1)," 2019. [Online]. Available: https://cwe.mitre.org/data/definitions/98.html. [Accessed: 17-May-2021].
[37] Santos O. Network security with NetFlow and IPFIX: big data analytics for information security. 2nd ed. Indianapolis: Cisco Press; 2016.

[38] A. Habibi Lashkari, G. Draper Gil, M. Mamun, A. Ghorbani, Characterization of Tor Traffic using Time based Features. 2017
[39] NetSA, "CERT NetSA Security Suite." [Online]. Available: https://tools.netsa.cert.org/. [Accessed: 26-Mar-2021].
[40] Carnegie Mellon, "YAF — Documentation." [Online]. Available: https://tools.netsa.cert.org/yaf/docs.html. [Accessed: 26-May-2021].
[41] J. Patterson and A. Gibson, *Deep learning : a practitioner's approach*, 1st ed. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2017.
[42] Atli BG. Anomaly-Based Intrusion Detection by Modeling Probability Distributions of Flow Characteristics. Aalto University; 2017.
[43] CIC, "IDS 2018|Datasets|Research|Canadian Institute for Cybersecurity|UNB," 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html. [Accessed: 17-Feb-2021].
[44] Čeponis Dainis, Goranin Nikolaj. Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset. Baltic J Modern Comput 2018;6(3). doi: https://doi.org/10.22364/bjmc10.22364/bjmc.6.310.22364/bjmc.2018.6.3.01.
[45] Goodfellow I, Bengio Y, Courville A. Deep learning, Vol. 1. USA: MIT Press Massachusetts; 2017.
[46] Sehgal S, Singh H, Agarwal M, Bhasker V, Shantanu,. Data analysis using principal component analysis. In: 2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom). p. 45–8.
[47] D. J. F. Wiley, "Chapter (5): Tarining deep prediction models," *R Deep Learning Essentials. Packt Publishing Ltd*, p. 98.
[48] N. Kang, "Multi-Layer Neural Networks with Sigmoid Function— Deep Learning for Rookies (2)," *Data Science*, 2017. [Online]. Available: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f. [Accessed: 17-Apr-2021].
[49] H. Mahmood, "The Softmax Function, Simplified," *Data Science*, 2018. [Online]. Available: The Softmax Function, Simplified. How a regression formula improves... %7C by Hamza Mahmood %7C Towards Data Science. [Accessed: 17-Apr-2021].
[50] G. Seif, "Understanding the 3 most common loss functions for Machine Learning Regression," *Data Science*, 2019. [Online]. Available: Understanding the 3 most common loss functions for Machine Learning Regression %7C by George Seif %7C Towards Data Science. [Accessed: 17-Apr-2021].
[51] J. Cao, K. Mao, J. Wu, and A. Lendasse, *Proceedings of ELM-2015 Volume 1: Theory, Algorithms and Applications (I)*, vol. 6. Springer, 2015
[52] A. Kalliola *et al.*, "Learning Flow Characteristics Distributions with ELM for Distributed Denial of Service Detection and Mitigation BT - Proceedings of ELM-2016," in *In ELM-2016 Cham-Switzerland*, 2018, pp. 129–143.
[53] Kanimozhi V, Jacob TP. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. In: 2019 international conference on communication and signal processing (ICCSP). p. 33–6.