IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 27, 2018, accepted January 3, 2019, date of current version April 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2895334

Deep Learning Approach for Intelligent Intrusion Detection System

R. VINAYAKUMAR[1], MAMOUN ALAZAB[2], (Senior Member, IEEE), K. P. SOMAN[1], PRABAHARAN POORNACHANDRAN[3], AMEER AL-NEMRAT[4], AND SITALAKSHMI VENKATRAMAN[5]

[1]Center for Computational Engineering and Networking, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore 641112, India
[2]College of Engineering, IT & Environment, Charles Darwin University, Casuarina, NT 0810, Australia
[3]Centre for Cyber Security Systems and Networks, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India
[4]School of Architecture, Computing, and Engineering, University of East London, London E16 2RD, U.K.
[5]Department of IT, Melbourne Polytechnic, Prahran Campus, Melbourne, VIC 3181, Australia

Corresponding author: R. Vinayakumar (vinayakumarr77@gmail.com)

This work was supported in part by the Paramount Computer Systems, in part by the Lakhshya Cyber Security Labs, and in part by the Department of Corporate and Information Services, Northern Territory Government of Australia.

ABSTRACT Machine learning techniques are being widely used to develop an intrusion detection system (IDS) for detecting and classifying cyberattacks at the network-level and the host-level in a timely and automatic manner. However, many challenges arise since malicious attacks are continually changing and are occurring in very large volumes requiring a scalable solution. There are different malware datasets available publicly for further research by cyber security community. However, no existing study has shown the detailed analysis of the performance of various machine learning algorithms on various publicly available datasets. Due to the dynamic nature of malware with continuously changing attacking methods, the malware datasets available publicly are to be updated systematically and benchmarked. In this paper, a deep neural network (DNN), a type of deep learning model, is explored to develop a flexible and effective IDS to detect and classify unforeseen and unpredictable cyberattacks. The continuous change in network behavior and rapid evolution of attacks makes it necessary to evaluate various datasets which are generated over the years through static and dynamic approaches. This type of study facilitates to identify the best algorithm which can effectively work in detecting future cyberattacks. A comprehensive evaluation of experiments of DNNs and other classical machine learning classifiers are shown on various publicly available benchmark malware datasets. The optimal network parameters and network topologies for DNNs are chosen through the following hyperparameter selection methods with KDDCup 99 dataset. All the experiments of DNNs are run till 1,000 epochs with the learning rate varying in the range [0.01–0.5]. The DNN model which performed well on KDDCup 99 is applied on other datasets, such as NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017, to conduct the benchmark. Our DNN model learns the abstract and high-dimensional feature representation of the IDS data by passing them into many hidden layers. Through a rigorous experimental testing, it is confirmed that DNNs perform well in comparison with the classical machine learning classifiers. Finally, we propose a highly scalable and hybrid DNNs framework called scale-hybrid-IDS-AlertNet which can be used in real-time to effectively monitor the network traffic and host-level events to proactively alert possible cyberattacks.

INDEX TERMS Cyber security, intrusion detection, malware, big data, machine learning, deep learning, deep neural networks, cyberattacks, cybercrime.

I. INTRODUCTION

Information and communications technology (ICT) systems and networks handle various sensitive user data that are prone by various attacks from both internal and external intruders [1]. These attacks can be manual and machine generated, diverse and are gradually advancing in obfuscations resulting in undetected data breaches. For instance, the Yahoo data breach had caused a loss of $350 M and Bitcoin breach resulted in a rough estimate of $70M loss [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Arshad.

Such cyberattacks are constantly evolving with very sophisticated algorithms with the advancement of hardware, software, and network topologies including the recent developments in the Internet of Things (IoT) [4]. Malicious cyberattacks pose serious security issues that demand the need for a novel, flexible and more reliable intrusion detection system (IDS). An IDS is a proactive intrusion detection tool used to detect and classify intrusions, attacks, or violations of the security policies automatically at network-level and host-level infrastructure in a timely manner. Based on intrusive behaviors, intrusion detection is classified into network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS) [5]. An IDS system which uses network behavior is called as NIDS. The network behaviors are collected using network equipment via mirroring by networking devices, such as switches, routers, and network taps and analyzed in order to identify attacks and possible threats concealed within in network traffic. An IDS system which uses system activities in the form of various log files running on the local host computer in order to detect attacks is called as HIDS. The log files are collected via local sensors. While NIDS inspects each packet contents in network traffic flows, HIDS relies on the information of log files which includes sensors logs, system logs, software logs, file systems, disk resources, users account information and others of each system. Many organizations use a hybrid of both NIDS and HIDS.

Analysis of network traffic flows is done using misuse detection, anomaly detection and stateful protocol analysis. Misuse detection uses predefined signatures and filters to detect the attacks. It relies on human inputs to constantly update the signature database. This method is accurate in finding the known attacks but is completely ineffective in the case of unknown attacks. Anomaly detection uses heuristic mechanisms to find the unknown malicious activities. In most of the scenarios, anomaly detection produces a high false positive rate [5]. To combat this problem, most organizations use the combination of both the misuse and anomaly detection in their commercial solution systems. Stateful protocol analysis is most powerful in comparison to the aforementioned detection methods due to the fact that stateful protocol analysis acts on the network layer, application layer and transport layer. This uses the predefined vendors specification settings to detect the deviations of appropriate protocols and applications. Though deep learning approaches are being considered more recently to enhance the intelligence of such intrusion detection techniques, there is a lack of study to benchmark such machine learning algorithms with publicly available datasets. The most common issues in the existing solutions based on machine learning models are: firstly, the models produce high false positive rate [3], [5] with wider range of attacks; secondly, the models are not generalizable as existing studies have mainly used only a single dataset to report the performance of the machine learning model; thirdly, the models studied so far have completely unseen today's huge network traffic; and finally the solutions are required to persevere today's rapidly increasing high-speed network size, speed and dynamics. These challenges form the prime motivation for this work with a research focus on evaluating the efficacy of various classical machine learning classifiers and deep neural networks (DNNs) applied to NIDS and HIDS. This work assumes the following;

- An attacker aims at pretence as normal user to remain hidden from the IDS. However, the patterns of intrusive behaviors differ in some aspect. This is due to the specific objective of an attacker for example getting an unauthorized access to computer and network resources.
- The usage pattern of network resources can be captured, however the existing methods ends up in high false positive rate.
- The patterns of intrusions exist in normal traffic with a very low profile over long time interval.

Overall, this work has made the following contributions to the cyber security domain:

- By combining both NIDS and HIDS collaboratively, an effective deep learning approach is proposed by modeling a deep neural network (DNN) to detect cyberattacks proactively. In this study, the efficacy of various classical machine learning algorithms and DNNs are evaluated on various NIDS and HIDS datasets in identifying whether network traffic behavior is either normal or abnormal due to an attack that can be classified into corresponding attack categories.
- The advanced text representation methods of natural language processing (NLP) are explored with host-level events, i.e. system calls with the aim to capture the contextual and semantic similarity and to preserve the sequence information of system calls. The comparative performance of these methods is conducted with the ADFA-LD and ADFA-WD datasets.
- This study uses various benchmark datasets to conduct a comparative experimentation. This is mainly due to the reason that each dataset suffers from various issues such as data corruptions, traffic variety, inconsistencies, out of date and contemporary attacks.
- A scalable hybrid intrusion detection framework called SHIA is introduced to process large amount of network-level and host-level events to automatically identify malicious characteristics in order to provide appropriate alerts to the network admin. The proposed framework is highly scalable on commodity hardware server and by joining additional computing resources to the existing framework, the performance can be further enhanced to handle big data in real-time systems.

The code and detailed results are made publicly available [7] for further research. The remainder of the chapter is organized as follows. Section II discusses various stages of compromise according to attackers perspective. Section III discusses the related works of similar research work done to NIDS and HIDS. Information of scalable framework, the mathematical details of DNNs and text representation methods for intrusion detection is placed

in Section IV. Section V includes information related to major shortcomings of IDS datasets, problem formulation and statistical measures. Section VI includes description of datasets. Section VII and Section VIII includes experimental analysis and a brief overview of proposed system and architecture design. Section IX presents the experimental results. Conclusion, future work directions and discussions are placed in Section X.

II. STAGES OF COMPROMISE: AN ATTACKER'S VIEW

Mostly, intrusions are initiated by unauthorized users named as attackers. An attacker can attempt to access a computer remotely via the Internet or to make a service remotely unusable. Detection of intrusion accurately requires understanding the method to successfully attack a system. Generally, an attack can be classified into five phases. They are reconnaissance, exploitation, reinforcement, consolidation, and pillage. An attack can be detected during the first three phases however once it reaches the fourth or fifth phase then the system will be fully compromised. Thus, it is very difficult to distinguish between a normal behavior and an attack. During the reconnaissance phase, an attacker tries to collect information related to reachable hosts and services, as well as the versions of the operating systems and applications that are running. During the exploitation phase, an attacker utilizes a particular service with the aim to access the target computer. A service may be identified as abusing, subverting, or breaching. An abusing service includes stolen password or dictionary attacks and subversion includes an SQL injection. After an illegal forced entry to a system, an attacker follows camouflage activity and then installs supplementary tools and services to take advantage of the privileges gained during the reinforcement phase. Based on the misused user account, an attacker tries to gain full system access. Finally, an attacker utilizes the applications that are accessible from the available user account. An attacker obtains a complete control over the system in the consolidation phase and the installed backdoor which is used for communication purposes during the consolidation phase. The final phase is pillage where an attacker's possible malicious activities include theft of data and CPU time, and impersonation.

Since computers and networks are assembled and programmed by humans, there are possibilities for bugs in both the hardware and software. These human errors and bugs can lead to vulnerabilities [8]. Confidentiality, data integrity and availability are main pillars of information security. Authenticity and accountability are also plays an important role in information security. Generally attacks against the confidentiality addresses passive attacks for example eavesdropping, integrity addresses active attacks for example system scanning attacks i.e. 'Probe' and availability addresses the attacks related to making network resources down so these will be unavailable for normal users for example denial of service ('DoS') and distributed denial of service ('DDoS'). IDS systems have limited capability to detect attacks related to eavesdropping. 'Probe' attack can be launched over either over a network or locally within a system. Now an attack can be defined as a set of actions that potentially compromises the confidentiality, data integrity, availability, or any kind of security policy of a resource. Primarily, an IDS system aims at detecting all these types of attacks to prevent the computers and networks from malicious activities. In this work, we focus towards the categorization scheme as suggested by the DARPA Intrusion Detection Evaluation.

III. RELATED WORKS

The research on security issues relating to NIDS and HIDS exists since the birth of computer architectures. In recent days, applying machine learning based solutions to NIDS and HIDS is of prime interest among security researchers and specialists. A detailed survey on existing machine learning based solutions is discussed in detail by [5]. This section discusses the panorama of largest study to date that explores the field of machine learning and deep learning approaches applied to enhance NIDS and HIDS.

A. NETWORK-BASED INTRUSION DETECTION SYSTEMS (NIDS)

Commercial NIDS primarily use either statistical measures or computed thresholds on feature sets such as packet length, inter-arrival time, flow size and other network traffic parameters to effectively model them within a specific time-window [6]. They suffer from high rate of false positive and false negative alerts. A high rate of false negative alerts indicates that the NIDS could fail to detect attacks more frequently, and a high rate of false positive alerts means the NIDS could unnecessarily alert when no attack is actually taking place. Hence, these commercial solutions are ineffective for present day attacks.

Self-learning system is one of the effective methods to deal with the present day attacks. This uses supervised, semi-supervised and unsupervised mechanisms of machine learning to learn the patterns of various normal and malicious activities with a large corpus of Normal and Attack network and host-level events. Though various machine learning based solutions are found in the literature, the applicability to commercial systems is in early stages [9]. The existing machine learning based solutions outputs high false positive rate with high computational cost [3]. This is because machine learning classifiers learn the characteristic of simple TCP/IP features locally. Deep learning is a complex subnet of machine learning that learns hierarchical feature representations and hidden sequential relationships by passing the TCP/IP information on several hidden layers. Deep learning has achieved significant results in long standing Artificial intelligence (AI) tasks in the field of image processing, speech recognition, natural language processing (NLP) and many others [10]. Additionally, these performances have been transformed to various cyber security tasks such as intrusion detection, android malware classification, traffic analysis, network traffic prediction, ransomware detection, encrypted text categorization, malicious URL

detection, anomaly detection, and malicious domain name detection [11]. This work focuses towards analyzing the effectiveness of various classical machine learning and deep neural networks (DNNs) for NIDS with the publicly available network-based intrusion datasets such as KDDCup 99, NSL-KDD, Kyoto, UNSW-NB15, WSN-DS and CICIDS 2017.

A large study of academic research used the de facto standard benchmark data, KDDCup 99 to improve the efficacy of intrusion detection rate. KDDCup 99 was used for the third International Knowledge Discovery and Data Mining Tools Competition and the data was created as the processed form of tcpdump data of the 1998 DARPA intrusion detection (ID) evaluation network. The aim of the contest was to create a predictive model to classify the network connections into two classes: Normal or Attack. Attacks were categorized into denial of service ('DoS'), 'Probe', remote-to-local ('R2L'), user-to-root ('U2R') categories. The mining audit data for automated models for ID (MADAMID) was used as feature construction framework in KDDCup 99 competition [17]. MADAMID outputs 41 features: first 9 features are basic features of a packet, 10-22 are content features, 23-31 are traffic features, and 32-41 are host-based features. The choices of available datasets are: (1) full dataset and (2) complementary 10% data. The detailed evaluation results of KDDCup 98 and KDDCup 99 challenge was published in [3]. Totally, 24 entries were submitted in the KDDCup 98, in that 3 winning entries used variants of decision tree to whom they showed only the marginal statistics significance in performance. The 9th winning entry in the contest used the 1-nearest neighbor classifier. The first significant difference in performance was found between 17th and 18th entries. This inferred that the first 17 submissions method were robust and were profiled by [3]. The Third International Knowledge Discovery and Data Mining Tools Competition task remained as a baseline work and after this contest many machine learning solutions have been found. Most of the published results took only the 10% data of training and testing and few of them used custom-built datasets. Recently, a comprehensive literature survey on machine learning based ID with KDDCup 99 dataset was conducted [18].

After the challenge, most of the published results of KDDCup 99 have used several feature engineering methods for dimensionality reduction [18]. While few studies employed custom-built datasets, majority used the same dataset for newly available machine learning classifiers [18]. These published results are partially comparable to the results of the KDDCup 99 contest.

In [19], the classification model consists of two-stages: i) P-rules stage to predict the presence of the class, and ii) N-rules stage to predict the absence of the class. This performed well in comparison with the aforementioned KDDCup 99 results except for the user-to-root ('U2R') category. In [20], the significance of feature relevance analysis was investigated for IDS with the most widely used dataset, KDDCup 99. For each feature they were able to express the feature relevance in terms of information gain. In addition, they presented the most relevant features for each class label. Reference [21] discussed random forest techniques in misuse detection by learning patterns of intrusions, anomaly detection with outlier detection mechanism, and hybrid detection by combining both the misuse and anomaly detection. They reported that the misuse approach worked better than winning entries of KDDCup 99 challenge results, and in addition anomaly detection worked better compared to other published unsupervised anomaly detection methods. Overall, it was concluded that the hybrid system enhances the performance with the advantage of combining both the misuse and anomaly detection approaches [22], [23], [72]. In [24], an ID algorithm using AdaBoost technique was proposed that used decision stumps as weak classifiers. Their system performed better than other published results with a lower false alarm rate, a higher detection rate, and a computationally faster algorithm. However, the drawback is that it failed to adopt the incremental learning approach. In [25], the performance of the shared nearest neighbor (SNN) based model in ID was studied and reported as the best algorithm with a high detection rate. With the reduced dataset they were able to conclude that SNN performed well in comparison to the K-means for 'U2R' attack category. However, their work failed to show the results on the entire testing dataset.

In [26], Bayesian networks for ID was explored using Naive Bayesian networks with a root node to represent a class of a connection and leaf nodes to represent features of a connection. Later, [27] investigated the application of Naive Bayes network to ID and through detailed experimental analysis, they showed that Bayesian networks performed equally well and sometimes even better in 'U2R' and 'Probe' categories in comparison with the winning entries of KDDCup 99 challenge. In [28], a non-parametric density estimation method based on Parzen-window estimators was studied with Gaussian kernels and Normal distribution. Without the intrusion data, their system was comparatively favorable to the existing winning entries that was based on ensemble of decision trees. In [29], a genetic algorithm based NIDS was proposed that facilitates to model both temporal and spatial information to identify complex anomalous behavior. An overview of ensemble learning techniques for ID was given in [30], and swarm intelligence techniques for ID using ant colony optimization, ant colony clustering and particle swarm optimization of systems were studied in [31]. A comparative study in such research works show that the descriptive statistics was predominantly used.

Overall, a comprehensive literature review shows very few studies use modern deep learning approaches for NIDS and the commonly used benchmark datasets for experimental analysis are KDDCup 99 and NSL-KDD [3], [32]–[34]. The IDS based on recurrent neural network (RNN) outperformed other classical machine learning classifiers in identifying intrusion and intrusion type on the NSL-KDD dataset [32]. Two level approach proposed for IDS in which the first level extracts the optimal features using sparse

autoencoder in an unsupervised way and classified using softmax regression [33]. The application of stacked autoencoder was proposed for optimal feature extraction in an unsupervised way where the proposed method is completely non-symmetric and classification was done using Random forest. Novel long short-term memory (LSTM) architecture was proposed and by modeling the network traffic information in time series obtained better performance. The proposed method performed well compared to all the existing methods and as well as KDDCup 98 and 99 challenge entries [3]. The performance of various RNN types were evaluated by [34]. Various deep learning architectures and classical machine learning algorithms were evaluated for anomaly based ID on NSL-KDD dataset [74]. The configuration of SVM was formulated as bi-objective optimization problem and solved using hyper-heuristic framework. The performance was evaluated for malware and anomaly ID. The proposed framework is very suitable for big data cyber security problems [75]. To enhance the anomaly based ID rate, the spatial and temporal features were extracted using convolutional neural network and long short-term memory architecture. The performance was shown on both KDDCup 99 and ISCX 2012 datasets [76]. Two step attack detection method was proposed along with a secure communication protocol for big data systems to identify insider attack. In the first step, process profiling was done independently at each node and in second step using hash matching and consensus, process matching was done [77]. An online detection and estimation method was proposed for smart grid system [78]. The method specifically designed for identifying false data injection and jamming attacks in real-time and additionally provides online estimates of the unknown and time-varying attack parameters and recovered state estimates [78]. A scalable framework for ID over vehicular ad hoc network was proposed. The framework uses distributed machine learning i.e. alternating direction method of multipliers (ADMM) to train a machine learning model in a distributed way to learn whether an activity normal or attack [79].

B. HOST-BASED INTRUSION DETECTION SYSTEMS (HIDS)

Various software tools such as Metasploit, Sqlmap, Nmap, Browser Exploitation provide the necessary framework to examine and gather information from target system vulnerabilities. Malicious attackers use such information to launch attacks to various applications like FTP server, web server, SSH server, etc. Existing methods such as firewall, cryptography methods and authentications aim to defend host systems against such attacks. However, these solutions have limitations and malicious attackers are able to gain unauthorized access to the system. To address this, a typical HIDS operates at host-level by analyzing and monitoring all traffic activities on the system application files, system calls and operating system [73]. These types of traffic activities are typically called as audit trials. A system call of an operating system is a key feature that interacts between the core kernel functions and low level system applications. Since an application makes communication with the operating system via system calls, their behavior, ordering, type and length generates a unique trace. This can be used to distinguish between the known and unknown applications [12]. System calls of normal and intrusive process are entirely different. Thus analysis of those system calls provides significant information about the processes of a system. Various feature engineering approaches have been used for system call based process classification. They are N-gram [12], [13], sequence gram [14] and pair gram [15]. An important advantage of HIDS is that it provides detailed information about the attacks.

The three main components of HIDS, namely the data source, the sensor, and the decision engine play an important role in detecting security intrusions. The sensor component monitors the changes in data source, and the decision engine uses the machine learning module to implement to the intrusion detection mechanism. However, the benchmarking the data source component requires much investigation.

Compiling the KDDCup 99 dataset involved the data source component with system calls and Sequence Time-Delay Embedding (STIDE) approach used to analyze the fixed length pattern of system calls to distinguish between normal and anomalous behaviors [13]. A large number of decision engines have been used to analyze patterns of system calls to detect intrusions. Such a data source is most commonly used among cyber security research community. Apart from system calls, since Windows operating system (OS) does not provide a direct access to system calls, log entries [35] and registry entry manipulations [36] form the other two most commonly used data sources. This work focuses on the decision engine component to benchmark the data source.

Classical methods aim to find information about the nature of the host activity by analyzing the patterns in the sequence of system calls. While STIDE was most commonly used simple algorithm, Support Vector Machines (SVMs), Hidden Markov Models (HMMs) and Artificial Neural Networks (ANNs) are more recently adopted complex methods. In [37], N-gram feature extraction approach was used for compiling the ADFA-LD system call data and N-gram features were passed to different classical machine learning classifiers to identify and categorize attacks. In [39], in order to reduce the dimensions of system calls, K-means and KNN were experimented using a frequency based model. A revised version of N-gram model was used in [38] to represent system calls with various classical machine learning classifiers for both Binary and Multi-class categories. An approach for HIDS based on N-gram system call representations with various classical machine learning classifiers was proposed in [40]. To reduce the dimensions of N-gram, dimensionality reduction methods were employed. In [41], frequency distribution based feature engineering approach with machine learning algorithms was explored to handle the zero-day and stealth attacks in Windows OS. In [42], an ensemble approach for HIDS was proposed using language modeling to reduce the false alarm rates which is a drawback in classical methods.

This method leveraged the semantic meaning and communications of system call. The effectiveness of their methods was evaluated on three different publicly available datasets.

Overall, the published results are limited in detecting the intrusions and cyberattacks using HIDS. Studies that show an increase in detection rate of intrusions and cyberattacks also show an increase in false alarm rate.

The pros and cons of NIDS and HIDS with its efficacy are discussed in detail by [16]. Major advantages of HIDSs are: HIDSs facilitate to detect local attacks and are unaffected by the encryption of network traffic. Major disadvantage is that they need all the configuration files to identify attack, but it is a daunting task due to the huge amount of data. Allowing access to big data technology in the domain of cyber security is of paramount importance, particularly IDS. The motivation of this research is to develop a novel scalable platform with hybrid framework of NIDS and HIDS, which is capable of handling large amount of data with the aim to detect the intrusions and cyberattacks more accurately.

IV. PROPOSED SCALABLE FRAMEWORK

Today's ICT system is considerably more complex, connected and involved in generating extremely large volume of data, typically called as big data. This is primarily due to the advancement in technologies and rapid deployments of large number of applications. Big data is a buzzword which contains techniques to extract important information from large volume of data. Allowing access to big data technology in the domain cyber security particularly IDS is of paramount importance [44]. The advancement in big data technology facilitates to extract various patterns of legitimate and malicious activities from large volume of network and system activities data in a timely manner that in turn facilitates to improve the performance of IDS. However, processing of big data by using the conventional technologies is often difficult [43]. The purpose of this section is to describe the computing architecture and the advanced methods adopted in the proposed framework, such as text representation methods, deep neural networks (DNNs) and the training mechanisms employed in DNNs.

A. SCALABLE COMPUTING ARCHITECTURE

The technologies such as Hadoop Map reduce and Apache Spark in the field of high performance computing is found to be an effective solution to process the big data and to provide timely actions. We have developed scalable framework based on big data techniques, Apache Spark cluster computing platform [45]. Due to the confidential nature of the research, the scalable framework details cannot be disclosed. The Apache spark cluster computing framework is setup over Apache Hadoop Yet Another Resource Negotiator (YARN). This framework facilitates to efficiently distribute, execute and harvest tasks. Each system has specifications(32 GB RAM, 2 TB hard disk, Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz) running over 1 Gbps Ethernet network.

The proposed scalable architecture employs distributed and parallel machine learning algorithms with various optimization techniques that makes it capable of handling very high volume of network and host-level events. The scalable architecture also leverages the processing capability of the general purpose graphical processing unit (GPGPU) cores for faster and parallel analysis of network and host-level events. The framework contains two types of analytic engines, they are real-time and non-real-time. The purpose of analytic engine is to monitor network and host-level events to generate an alert for an attack. The developed framework can be scaled out to analyze even larger volumes of network event data by adding additional computing resources. The scalability and real-time detection of malicious activities from early warning signals makes the developed framework stand out from any system of similar kind.

B. TEXT REPRESENTATION METHODS

System calls are essential in any operating system depicting the computer processes and they constitute a humongous amount of unstructured and fragmented texts that a typical HIDS uses to detect intrusions and cyberattacks. In this research we consider text representation methods to classify the process behaviors using system call trace. Classical machine learning approaches adopt feature extraction, feature engineering and feature representation methods. However, with advanced machine learning embedded approach such as deep learning, the necessity of the feature engineering and feature extraction steps can be completely avoided. We adopt such advanced deep learning along with text representation methods to capture the contextual and sequence related information from system calls. The following feature representation methods in the field of NLP are used to convert the system calls into feature vectors in this study.

- **Bag-of-Words (BoW):** This classical and most commonly used representation method is used to form a dictionary by assigning a unique number for each system call. Term document matrix (TDM) and term frequency-inverse document frequency (TF-IDF) are employed to estimate the feature vectors. The drawback is that it cannot capture the sequence information of system calls [46].
- **N-grams:** An N-gram text representation method has the capability to preserve the sequence information of system calls. The size of N can be 1 (uni-gram), 2 (bi-gram), 3 (tri-gram), 4 (four-gram), etc., which can be employed appropriately depending on the context.
- **Keras Embedding:** This follows a sequential representation method to convert the system calls into a numeric form of vocabulary by simply assigning a unique number for each system call. The size of vocabulary defines the number of unique system calls and their frequency of occurrence places them in an ascending order within a lookup table. Each system call in a vector is transformed to a numeric using the lookup table for assigning a

corresponding index. We adopt a fixed length vector method by transforming all vectors to the same length.

C. DEEP NEURAL NETWORK (DNN)

We employ an artificial neural network (ANN) approach as the computational model since it is influenced by the characteristics of biological neural networks to incorporate intelligence in our proposed method. Feed forward neural network (FFN), a type of ANN is represented as a directed graph to pass various system information along edges from one node to another without forming a cycle. We adopt a multilayer perceptron (MLP) model which is a type of FFN having three or more layers with one input layer, one or more hidden layers and an output layer in which each layer has many neurons or units in mathematical notation. We select the number of hidden layers by following a hyper parameter selection method. The information is transformed from one layer to another layer in a forward direction with neurons in each layer being fully connected. MLP is defined mathematically as $O : \mathbb{R}^m \times \mathbb{R}^n$ where m is the size of the input vector $x = x_1, x_2, \cdots, x_{m-1}, x_m$ and n is the size of the output vector $O(x)$ respectively. The computation of each hidden layer h_i is mathematically defined as

$$h_i(x) = f({w_i}^T x + b_i) \tag{1}$$

where $h_i : \mathbb{R}^{d_i-1} \rightarrow \mathbb{R}^{d_i}, f : \mathbb{R} \rightarrow \mathbb{R}, w_i \in \mathbb{R}^{d \times d_{i-1}}, b \in \mathbb{R}^{d_i}$, d_i denotes the size of the input, f is the non-linear activation function, which is either a *sigmoid* (values in the range [0, 1]) or a *tangent* function (values in the range [1, −1]). For the classification problem of Multi-class, our MLP model uses *softmax* function as the non-linear activation function. *softmax* function outputs the probabilities of each class and selects the largest value among probability values to give a more accurate value. The mathematical formulae for *sigmoid*, *tangent* and *softmax* activation function are given below.

$$sigmoid = \frac{1}{1 + e^{-x}} \tag{2}$$

$$\tan gent = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3}$$

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{4}$$

where x defines an input.

Three-layer MLP with a *softmax* function in output layer is same as a Multi-class logistic regression model. In general terms, for many hidden layers, MLP is formulated as follows:

$$H(x) = H_l(H_{l-1}(H_{l-2}(\cdots(H_1(x))))) \tag{5}$$

This way of stacking hidden layers is typically called deep neural networks (DNNs). The architecture of deep neural network (DNN) as shown in Figure 1 contains 1 hidden layer. It takes inputs $x = x_1, x_2, \cdots, x_{m-1}, x_m$ and outputs $o = o_1, o_2, \cdots, o_{c-1}, o_c$. However, all the connections and hidden layers along with its units are not shown in Figure 1.

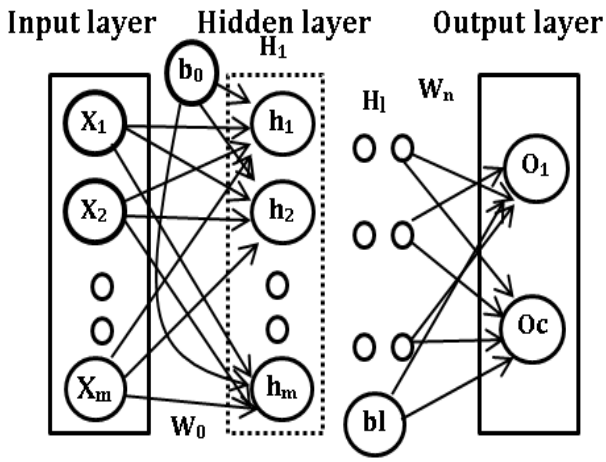


FIGURE 1. Architecture of a deep neural network (DNN).

We employ DNNs as a more advanced model of the classical FFN with each hidden layer using the non-linear activation function, *ReLU* as it helps to reduce the state of vanishing and error gradient issue [47]. The advantage of *ReLU* is that it is faster than other non-linear activation functions and facilitates training the MLP model with the large number of hidden layers. The hidden layers define the depth of the neural network and the maximum neurons define the width of the neural network.

The uniqueness of our method is in modeling the loss functions and the *ReLU* to maximize deep learning efficiently. These are described in detail.

1) Loss functions: In modeling an MLP, finding an optimal parameter is essential towards achieving good performance. This includes the loss function as an initial step. A loss function is used to calculate the amount of difference between the predicted and target values. This is defined mathematically as:

$$d(t, p) = \| t - p \|_2^2 \tag{6}$$

where t denotes the target value and p denotes the predicted value.
Multi-class classification uses the negative log probability with t as the target class and $p(pad)$ as the probability distributions as represented below:

$$d(t, p(pd)) = -\log p(pd)_t \tag{7}$$

However, the network receives a list of corrected input-output set $i\_o = (i_1, o_1), (i_2, o_2), \cdots, (in, on)$ in the training process. Then, we aim to decrease the mean of losses as defined below:

$$loss(in, on) = \frac{1}{n} \sum_{i=1}^{n} d(oil, f(iris)) \tag{8}$$

The loss function has to be minimized to get better results in a neural network. A loss functions is

defined as;

$$Train_{i\_o}(\theta) \equiv L_{i\_o}(\theta) = \frac{1}{n}\sum_{i=1}^{n} d(o_i, f_\theta(i_i)) \quad (9)$$

where $\theta = (w_1, b_1, \cdots, w_n, b_n)$
Loss function minimization $L_{i\_o}(\theta)$ is done by following a right selection of the value $\theta \in \mathbb{R}^d$ and inherently includes the estimation of $f_\theta(p_i)$ and $\nabla f_\theta(i_i)$ at the cost $|i\_o|$

$$\min_{\theta} \; L(\theta) \quad (10)$$

Various optimization techniques exist and we adopt Gradient descent as it is most commonly used. Gradient descent uses the following rule to calculate and update parameter repeatedly:

$$\theta^{new} = \theta_{old} - \alpha \nabla_\theta L(\theta) \quad (11)$$

where α denotes learning rate and it is selected based on a hyper parameter selection approach. To find a derivative of L, backpropagation or backward propagation of errors algorithm is adopted. Backpropagation uses chain rule to compute $\theta \in \mathbb{R}^d$ with the aim to minimize the loss function $L_{i\_o}(\theta)$. However, most of the neural network uses an extension of backpropagation called as stochastic gradient descent (SGD) for finding minimum θ. SGD uses a mini batch of training samples *im_om*, in which training samples *i_o* are chosen randomly instead of using the entire training set $im\_om \subseteq i\_o$. SGD update rule is given as:

$$\theta^{new} = \theta_{old} - \alpha \nabla_\theta J(\theta; im^{(i)}, om^{(i)}) \quad (12)$$

where $im^{(i)}, om^{(i)}$ denotes input-output pair training samples.

2) Rectified Linear Unit (*ReLU*): Rectified linear unit (*ReLU*) is found to have a great proficiency and has the tendency to accelerate the training process [47]. *ReLU* was the main breakthrough in the neural network history for reducing the vanishing and exploding gradient issue. It's found as the most efficient method in terms of time and cost for training huge data in comparison to the classical non-linear activation function such as *sigmoid* and *tangent* function [47]. We refer to neurons with this non linearity following [47]. The mathematical formula for *ReLU* is defined as follows

$$f(x) = \max(0, x) \quad (13)$$

where x defines input.

V. PROBLEM FORMULATION, DATASET LIMITATIONS AND STATISTICAL MEASURES

A. PROBLEM FORMULATION FOR NIDS

Generally, the network traffic data is collected and stored in raw TCP dump format. Later, this data can be preprocessed and converted into connection records. A connection is simply a sequence of TCP packets starting and ending at well-defined times with well-defined protocols. Each connection record includes 100 bytes of information and labeled as either Normal or as an Attack with exactly one particular attack type. Each connection record has a vector and defined as follows

$$CV = (f_1, f_2, \cdots, f_n, cl) \quad (14)$$

where f denotes features of length n, values of each $f \in \mathbb{R}$ and cl denotes a class label.

B. PROBLEM FORMULATION FOR HIDS

In general, all the system events which are the system calls are collected for each process. Each process p is composed of sequence of system calls $S = sp_1, sp_2, \cdots sp_n$ where $sp \in S$, sp is a finite set of system calls and S is the set of system calls used by the host. A sequence of system call information is used to distinguish the behavior between the Normal and Attack categories. The sp along with the label such as Normal or Attack can be used to learn the behaviors of Normal and Attack activities.

C. DATASET LIMITATIONS

Most of the datasets which represents the current network traffic attacks are private due to privacy and security issues. On the other direction, the datasets which are publicly available are laboriously anonymized and suffer from various issues. In particular they failed to validate that their datasets typically exhibit the real-world network traffic profile. KDDCup 99 is one of the most commonly used publicly available datasets. Although with some known harsh criticisms, it has been continually used as an effective benchmark dataset for many of the research study towards NIDS over the years. In contrast to critiques of strategy to create dataset, [50] revealed the detailed analysis of the contents and located the non-uniformity and simulated artifacts in the simulated network traffic data. They strived to scale the performance of network anomaly detection between the KDDCup 99 and varied KDDCup 99. They reported that many of the network attributes particularly, remote client address, TTL, TCP options and TCP window size are indicated as small and limited range in KDDCup 99 datasets but actually exhibit to be of large and growing range in real world network traffic environment.

In [51], discussions indicate why the machine learning classifiers have limited capability in detecting the attacks that belong to content ('R2L' and 'U2R') category in KDDCup 99 dataset. With this dataset, none of the machine learning classifiers were able to improve the attack detection rate. They admitted that the possibility of getting high attack detection rate in most of the cases is by producing new dataset with a combination of training and testing datasets. In addition, [52] found that many 'snmpgetattack' belongs to 'R2L' category attacks. As a result, in most of the cases, machine learning classifier poorly performs with this data.

DARPA / KDDCup 88 failed to evaluate the classical IDS and it was one of the major of many criticisms. To mitigate

this [53] used Snort ID system on DARPA / KDDCup 98 tcpdump traces. The system performed poorly, the accuracy and the false positive rates were impermissible. This is mainly due to the system failed to detect the attacks belongs to the 'DoS' and 'Probe' category with a fixed signature. In contrast to this, the detection performance of 'R2L' and 'U2R' is much better.

Despite of the harsh criticisms yet, KDDCup 99 is been the most widely used reliable benchmark dataset in most of the study related to ID system evaluation and other security related tasks [55]. To resolve the inherent issues that are exists in KDDCup 99, [55] proposed a most refined version called NSL-KDD. They removed the redundant connection records in the entire train and test data and in addition the invalid records, numbered 136,489 and 136,497 were removed from test data. Thus it protects the classifier not to be biased in the direction of the more frequent connection records. NSL-KDD is also not a real world representative of network traffic data. Still, this refined version failed to entirely solve the issues reported by [57]. To enhance the performances in detecting attacks, 10 more extra features were added with 14 important features from KDDCup 99 [58]. The Kyoto dataset was generated using honeypots. Thus, each flow of network traffic was done automatically. The normal traffic of Kyoto dataset was not captured from the real world network traffic. Moreover, the dataset doesn't contain false positives that help to minimize the number of alerts to the network admin [58]. In [56] generated a new dataset following two different profile system in which one system was for generating attacks and other one was for normal activities. This dataset doesn't contain network traffic of HTTPS protocol. Most of the attacks were simulated and failed to preserve the characteristics of real world statistics. In [57] proposed UNSW-NB15. They adopted the notion of profiles that contains the comprehensive information of intrusions and applications, protocols, or lower level network entities from the modern network traffic and detailed information about the network traffic. Recently, to provide benchmark dataset to the research community, [59] generated reliable dataset. This meets the real world benign and attacks of network activities. Moreover, the detailed evaluation of network traffic features was done by them and detailed experiments towards the importance of features to detect various attacks were done.

Most widely used dataset for HIDS is KDDCup 98, KDDCup 99 and University of New Maxico (UNM). These datasets were compiled decades ago and most of them are irrelevant for today's operating system. Recently, [62] made the dataset to be publicly available. Thus, this dataset has been used as new benchmark for evaluating system call based HIDS. The dataset comprises of modern vulnerability exploits and attacks.

D. STATISTICAL MEASURES

In evaluation to estimate the various statistical measures the ground truth value is required. The ground truth composed of set of connection records labeled either Normal or Attack in the case of Binary classification. Let L and A be the number of Normal and Attack connection records in the test dataset, respectively and the following terms are used for determining the quality of the classification models:

- True Positive (TP) - the number of connection records correctly classified to the Normal class.
- True Negative (TN) - the number of connection records correctly classified to the Attack class.
- False Positive (FP) - the number of Normal connection records wrongly classified to the Attack connection record.
- False Negative (FN) - the number of Attack connection records wrongly classified to the Normal connection record.

Based on the aforementioned terms, the following most commonly used evaluation metrics are considered.

1) **Accuracy:** It estimates the ratio of the correctly recognized connection records to the entire test dataset. If the accuracy is higher, the machine learning model is better ($Accuracy \in [0, 1]$). accuracy serves as a good measure for the test dataset that contains balanced classes and defined as follows

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

2) **Precision:** It estimates the ratio of the correctly identified attack connection records to the number of all identified attack connection records. If the Precision is higher, the machine learning model is better ($Precision \in [0, 1]$). Precision is defined as follows

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

3) **F1-Score:** F1-Score is also called as F1-Measure. It is the harmonic mean of Precision and Recall. If the F1-Score is higher, the machine learning model is better ($F1{-}Score \in [0, 1]$). F1-Score is defined as follows

$$F1 - Score = 2 \times \left(\frac{\Pr ecision \times Recall}{\Pr ecision + Recall} \right) \tag{17}$$

4) **True Positive Rate (TPR):** It is also called as Recall. It estimates the ratio of the correctly classified Attack connection records to the total number of Attack connection records. If the TPR is higher, the machine learning model is better ($TPR \in [0, 1]$). TPR is defined as follows

$$TPR = \frac{TP}{TP + FN} \tag{18}$$

5) **False Positive Rate (FPR):** It estimates the ratio of the Normal connection records flagged as Attacks to the total number of Normal connection records. If the FPR is lower, the machine learning model is better ($FPR \in [0, 1]$). FPR is defined as follows

$$FPR = \frac{FP}{FP + TN} \tag{19}$$

TABLE 1. Training and testing connection records from KDDCup 99 and NSL-KDD datasets.

Attack category	Description	Data instances - 10 % data			
		KDDCup 99		NSL-KDD	
		Train	Test	Train	Test
Normal	Normal connection records	97,278	60,593	67,343	9,710
DoS	Attacker aims at making network resources down	391,458	229,853	45,927	7,458
Probe	Obtaining detailed statistics of system and network configuration details	4,107	4,166	11,656	2,422
R2L	Illegal access from remote computer	1,126	16,189	995	2,887
U2R	Obtaining the root or super-user access on a particular computer	52	228	52	67
Total		**494,021**	**311,029**	**125,973**	**22,544**

6) **Receiver Operating Characteristics (ROC) curve:** ROC is plotted based on the trade-off between the *TPR* on the y axis to *FPR* on the x axis across different thresholds. Area Under the ROC Curve (*AUC*) is the size of the area under the ROC curve used along with ROC as a comparison metric for the machine learning models. If the *AUC* is higher, the machine learning model is better.

$$AUC = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP}$$

VI. MODELLING THE DATASET

Due to security and privacy issues, most of the datasets are not publicly available. Additionally, the data which are publicly available are laboriously anonymized and do not contemplate today's network traffic variety. Due to these issues, the exemplary dataset is yet to be discerned [64]. The details of various IDS datasets are discussed in [59] and [66]. A detailed overview of available datasets between 1998 and 2016 is discussed in detail by [66]. We consider the pros and cons of existing datasets used in NIDS and HIDS and discuss how our datasets were modeled.

A. DATASETS USED IN NIDS

1) **KDDCup 99:** KDDCup 99 dataset was built by processing tcpdump data of the 1998 DARPA intrusion detection challenge dataset. The Mining Audit data for automated models for ID (MADMAID) framework was used to extract features from raw tcpdump data. The detailed statistics of the dataset is reported in Table 1. KDDCup 1998 dataset was created by MIT Lincon laboratory using 1000's of UNIX machines and 100's users accessing those machines. The network traffic data was captured and stored in tcpdump format for 10 weeks. The data of first seven weeks was used as training dataset and rest used as testing dataset. KDDCup 99 dataset is available in two forms. They are full dataset and 10% dataset. The dataset contains 41 features and 5 classes ('Normal', 'DoS', 'Probe', 'R2L', 'U2R'). These features are grouped into different categories as given below:

- Basic features [1-9]: The packet capture (Pcap) files of tcpdump are used to extract the basic features from the packet headers, TCP segments, and UDP datagram instead of payload. This task was carried out using a remodeled network analysis framework, Bro IDS.
- Content features [10-22]: Content features are extracted from the full payload of TCP/IP packets rooted on domain knowledge in tcpdump files. The feature analysis of payload has remained as research area for the last years. Recently, in [65], a deep learning approach was introduced to analyze the entire payload data instead of following the feature extraction process. Content features are mainly used to identify 'R2L' and 'U2R' category attacks. For example, many failed login attempts is the most prominent feature to indicate the malicious behavior in the entire payload. Unlike other category attacks, 'R2L' and 'U2R' category do not have the prominent sequential patterns due to events happening in a single connection.
- Time-based traffic features [23-41]: Time-based traffic features are extracted with a specific temporal window of two seconds. These are grouped into 'same host' and 'same service' based on the connection characteristics in the past 2 seconds. To handle slow probing attacks, the aforementioned characteristics are recalculated based on a connection window of 100 connections to the same host. These are typically termed as connection based or host-based traffic features.

2) **NSL-KDD:** NSL-KDD is the distilled version of KDDCup 99 intrusion data. The filters are used to remove redundant connection records in KDDCup 99 and connection records numbered 136,489 and 136,497 are removed from the test data. NSL-KDD can protect machine learning algorithms not to be biased. This can suits well for misuse detection in compared to the KDDCup 99 dataset. This also suffers from representing the real-time network traffic profile characteristics. The detailed statistics of NSL-KDD is reported in Table 1.

TABLE 2. Training and testing connection records of partial dataset of UNSW-NB15.

Class	Description	Train	Test
Normal	Normal connection records	56,000	37,000
Fuzzers	Attacks related to spams, html files penetrations and port scans	18,184	6,062
Analysis	Attacks related to port scan, html file penetrations and spam	2,000	677
Backdoors	Backdoors is a mechanism used to access a computer by evading the background existing security.	1,746	583
DoS	Intruder aims at making network resources down and consequently, resources are inaccessible to authorized users	12,264	4,089
Exploits	The security hole of operating system or the application software is understand by an attacker with the aim to exploit vulnerability	33,393	11,132
Generic	Attacks are related to block-cipher	40,000	18,871
Reconnaissance	A target system is observe by an attacker to gather information for vulnerability	10,491	3,496
Shell code	A small part of program termed as payload used in exploitation of software	1,133	378
Worms	Worms replicate themselves and distributed to other system through the computer network	130	44
Total		**93,500**	**28,481**

3) **UNSW-NB15:** The cyber security research team of Australian Centre for Cyber Security (ACCS) has introduced a new data called as UNSW-NB15 to resolve the issues found in the KDDCup 99 and NSL-KDD datasets. This data is generated in a hybrid way, containing the normal and attack behaviors of a live network traffic using IXIA Perfect Storm tool that has a repository of new attacks and common vulnerability exposures (CVE), a storehouse containing information regarding security vulnerabilities and exposures, which are known publicly. Two servers were used in IXIA traffic generator tool where one server generated the normal activities, whereas the other generated malicious activities in the network. Tcpdump tool used for capturing network packet traces which took several hours to compile the whole data of 100 GBs that were divided into 1,000 MB pcaps using tcpdump. From pcap files, the features were extracted using Argus and Bro-IDS in Linux Ubuntu 14.0.4. In addition to the above methods, depth analysis of each packet was done with 12 algorithms which are developed using C#. The data is accessible in two forms as follows:
   a) Full connection records consisting of 2 million connection records
   b) A partition of full connection records which is composed of 82,332 train connection records and 175,341 test connection records confined with 10 attacks. The partitioned dataset consists of 42 features with their parallel class labels which are Normal and nine different Attacks. The information regarding simulated attacks category and its detailed statistics are described in Table 2.
4) **Kyoto:** The honeypot systems of Kyoto university network traffic data have 24 statistical features. Among 24 features, 14 features are from KDDCup 99. These features are important as they are collected from raw traffic data of Kyoto university honeypot systems. Additionally, 10 more features are identified with the honeypot's network traffic system. In this work, the network logs of the year 2015 are considered. The logs are preprocessed and divided into training and testing datasets. The detailed statistics of the connection records are reported in Table 5.
5) **WSN-DS:** It is an IDS dataset developed for wireless sensor networks (WSN). This composed of four different types of 'DoS' attacks: Blackhole, Grayhole, Flooding, and Scheduling. They used Low-energy adaptive clustering hierarchy (LEACH) protocol to collect data from network Simulator 2 (NS-2) and then preprocessed to generate 23 features. This dataset was termed as WSN-DS and its detailed statistics is reported in Table 3.
6) **CICIDS2017:** This dataset includes the contemporary activities of benign and attacks which depicts the real-time network traffic. The main interest is given towards collecting the real-time background traffic during creating this dataset. Using B-profile system, benign background traffic was collected. This benign traffic contains the characteristics of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols. The network traffic for five days was collected and dumped with normal activity traffic on one day, and attacks injected on other days. The various attacks injected were Brute Force FTP, Brute Force SSH, 'DoS', Heartbleed, Web

TABLE 3. Training and testing WSN-DS dataset.

Class	Description	Train	Test
Normal	Normal connection records	238,046	102,020
Blackhole	It is a kind of 'DoS' attack where an attacker attacks LEACH protocol and during initial time itself they publicize themselves as a CH	7,033	3,015
Grayhole	It is a kind of 'DoS' attack where an attacker attacks LEACH protocol and during initial time itself they publicize themselves as a CH for other nodes	10,217	4,379
Flooding	Using different ways, an attacker attacks LEACH protocol	2,318	994
Scheduling	Scheduling attack happens during the setup phase of LEACH protocol	4,646	1,992
Total		**262,260**	**112,400**

TABLE 4. Training and testing CICIDS 2017 dataset.

Class	Description	Train	Test
Normal	Normal connection records	60,000	20,000
SSH-Patator	Secure shell - Representation of brute force attack	5,000	897
FTP-Patator	File transfer protocol - Representation of brute force attack	7,000	938
DoS	Intruder aims at making network resources down and consequently, resources are inaccessible to authorized users	6,000	2,000
Web	Attacks are related to web	2,000	180
Bot	Hosts are controlled by bot owners to perform various tasks such as steal data, send spam and others	1,500	466
DDoS	Distributed Denial of Service ('DDoS') is an attempt made to make services down using multiple sources. These are achieved using botnet	6,000	2,000
PortScan	Port scan is used to find the specific port which is open for a particular service. Using this attacker can get information related to sender and receiver's listening information	6,000	2,000
Total		**93,500**	**28,481**

Attack, Infiltration, Botnet and 'DDoS'. They also claimed that their dataset covers 11 important criteria which were discussed by [66]. The detailed information of CICIDS 2017 dataset is reported in Table 4.

We have randomly chosen 20,000 connection records in NIDS dataset and passed them into t-SNE [63] and their visual representations are given in Figure 2 for KDDCup 99 and Figure 3 for CICIDS 2017. Almost both the datasets are non-linearly separable and the connection records of CICIDS 2017 is considered as more complex in comparison with KDDCup 99. Also, CICIDS 2017 dataset is released recently and contain attacks which have occurred recently. Moreover, the CICIDS 2017 dataset has the characteristics of a real-time network traffic.

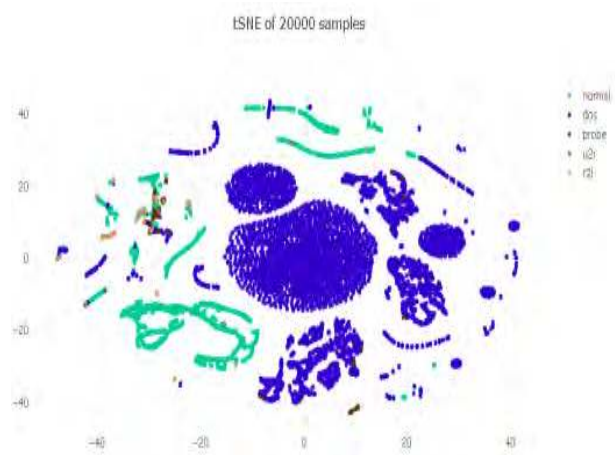


FIGURE 2. t-SNE visualization of KDDCup 99.

B. DATASETS USED IN HIDS

The windows and Linux operating system are the most well-known and most commonly used operating system (OS).

Most commonly used datasets for host-based intrusion detection are KDDCup 98, KDDCup 99[1] and UNM.[2] These datasets were compiled decades ago and do not include the attacks of modern computer systems. The issues of these datasets were discussed in detail by [62] and proposed a new dataset called as ADFA-LD and ADFA-WD.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://www.cs.unm.edu/ immsec/systemcalls.htm

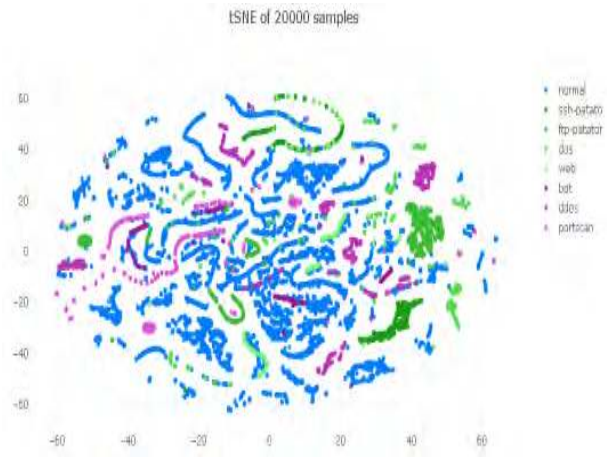


FIGURE 3. t-SNE visualization of CICIDS 2017.

TABLE 5. Training and testing Kyoto dataset.

Class	Training	Testing
Normal	2,384,645	1,405,391
Attack	670,037	158,532
Total	3,054,682	1,563,923

TABLE 6. ADFA-LD and ADFA-WD datasets.

Dataset	ADFA-LD		ADFA-WD	
	Traces	System calls	Traces	System calls
Train	833	308,077	355	13,504,419
Validation	4,372	2,122,085	1827	117,918,735
Attack	746	317,388	5,542	74,202,804
Total	5,951	2,747,550	7,724	205,625,958

1) ADFA Linux (ADFA-LD)/ADFA Windows(ADFA-WD): ADFA-LD is a sequence of system calls dataset which was collected in networks of modern operating systems [62]. These hosts were connected to a Linux local server. This comprised of several traces of system calls which were collected under different situations. This impersonated the real-time situations. These system call traces represented system level vulnerabilities and attacks. This server permitted many services such as remote access, web server, database on an operating system Ubuntu 11.04 (Linux kernel 2.6.38). The SSH, FTP and MySQL 14.14 services used their default ports. Apache 2.2.17 and PHP 5.3.5 and Tiki-Wiki 8.1 are installed as web based services and web based collaborative tools respectively. The detailed statistics of ADFA-LD is reported in Table 6. The attack types used in ADFA-LD dataset collection is reported in Table 7. The attack dataset of ADFA-LD is randomly split into 55% training and 45% testing. The normal traces of training and validation data are merged and randomly split for the 55% training and 45% testing.

TABLE 7. Types of attacks in ADFA-LD dataset.

Attack	Number of Traces
Adduser	91
Java-Meterpreter	124
Hydra-FTP	162
Hydra-SSH	176
Meterpreter	75
Web-Shell	118

The ADFA-WD comprises of system calls and dynamic link library (DLL) for various attacks. This was collected in Windows XP SP2 host using Procmon program [62]. The system setup enabled default firewall and Norton AV 2013. The system was open for file sharing and to run different applications like FTP server, streaming media server, database server, web server, PDF server etc. Using Metasploit framework and other custom approaches 12 different known vulnerabilities were exploited for installed applications. The detailed statistics of the ADFA-WD dataset is reported in Table 6.

VII. EXPERIMENTAL DESIGN

All the experiments were implemented using Python on an Ubuntu 14.0.4 LTS. All classical machine learning algorithms were implemented using Scikit-learn.[3] Deep neural networks (DNNs) was implemented using GPU enabled TensorFlow[4] as backend with Keras[5] higher level framework. The GPU was NVidia GK110BGL Tesla K40 and CPU had a configuration (32 GB RAM, 2 TB hard disk, Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz) running over 1 Gbps Ethernet network. To evaluate the performance of DNNs and various classical machine learning classifiers on various NIDS and HIDS datasets, the following different test cases were considered.

1) Classifying the network connection record as either benign or attack with all features.
2) Classifying the network connection record as either benign or attack and categorizing an attack into its categories with all features.
3) Classifying the network connection record as either benign or attack and categorizing an attack into its categories with minimal features.

A. FINDING OPTIMAL PARAMETERS IN DNNs

As DNNs are parametrized, the performance depends on the optimal parameters. The optimal parameter determination for DNNs network parameter and DNNs network topologies was done only for KDDCup 99 dataset. To identify the ideal

[3]https://scikit-learn.org/stable/
[4]https://www.tensorflow.org/
[5]https://keras.io/

parameter for the DNNs, a medium sized architecture was used for experiments with a specific hidden units, learning rate and activation function. A medium sized DNN contains 3 layers. One is input layer, second one is hidden layer or fully connected layer and third one is output layer. For KDDCup 99, the input layer contains 41 neurons, hidden layer contains 128, 256, 384, 512, 640, 768, 896 and 1,024 units and output layer contains 1 neuron in classifying the connection record as either normal or attack. It contains 5 neurons in classifying the connection record as either normal or attack and categorizing attack into corresponding attack categories. The connection between the units between input layer and hidden layer and hidden layer to output layer are fully connected. Initially, the train and test datasets were normalized using $L2$ normalization. Two trials of experiments were run for hidden units 128, 256, 384, 512, 640, 768, 896 and 1,024 with a medium sized DNN. The experiment was run for each parameter with appropriate units and for 300 epochs. The DNN with various units have learnt the patterns of normal connection records with epochs 200 in comparison to the those with attacks. To capture the significant features which can distinguish the attack connection record by DNN, 200 epochs were required. After 200 epochs, the performance of normal connection records fluctuated due to overfitting. Each number of units with DNN took varied number of epochs to attain considerable performance. It was found that the layer containing 1,024 units had shown highest number of attack detection rates. When we increased the number of hidden units from 1,024 to 2,048, the performance in attack detection rate deteriorated. Hence, we decided to use 1,024 units for the rest of the experiments. The medium sized DNN with 1,024 units in hidden layer was used for experiments with Multi-class classification of KDDCup 99 using three trials of experiments for each hidden units until 500 epochs as the DNN performed well in comparison to the other units. A medium sized DNN with less number of units, 128, 256 and 384 learned the patterns of high frequency attack, 'DoS'. The performance in detection of 'DoS' attack remained same for other variants in the number of units of a medium sized DNN. Due to the large number of connection records of 'DoS', DNN network with less number of units was able to achieve optimal detection rate. The acceptable detection rate for 'Probe' category of attacks was found with units 512 and 640. A medium sized DNN with hidden units of 896 performed well for detection of 'R2L' attacks in comparison to the other units. A medium sized DNN required 1,024 units to detect the attacks of 'U2R'. Once the number of units increased from 1,024, the performance of DNN network deteriorated. By considering all these test experiments, 1,024 was set as the ideal hidden layer units. To achieve a considerable performance, each DNN network topology required varied number of epochs. DNN network with less number of parameters have achieved good performance till 100 epochs but when it reaches 500 epochs, complex DNN networks have performed well in comparison to the DNN network with less number of parameters.

In order to find an optimal learning rate, three trials of experiments for 500 epochs with learning rate varying in the range [0.01-0.5] were run. These experiments used a medium sized DNN with 1,024 units. Learning rate has a strong impact on the training speed, thus we had selected the range [0.01-0.5]. The peak value for attack detection rate was obtained when the learning rate was 0.1. There was a quick decrease in attack detection rate when the learning rate was 0.2 and reached to peak accuracy at learning rates of 0.35, 0.45 and 0.45 compared to learning rate 0.1. This attack detection rate was intensified by running the experiments till 1,000 epochs. As we had examined more complex architectures for this experiment, it showed less performance for epochs less than 500, henceforth we decided to use the learning rate 0.1 for the rest of the experimentation process, because learning rate greater than 0.1 was found to be time consuming. To find the optimal learning rate for Multi-class classification of KDDCup 99, we run two trials of experiments for learning rate in the range [0.01-0.5]. The experiments with lower learning rate 0.01 showed better performance for attacks 'DoS' and 'Probe'. When we increase the learning rate from 0.01, the attack detection rate remained same. Experiments with learning rate 0.1 has showed optimal performance in detection of 'R2L' and 'U2R' attacks category. Based on the observations of performance of detection of attacks in both Binary and Multi-class classification, the learning rate was set to 0.1

In third trial of experiments, we had also run experiments with the *sigmoid* and *tanh* activation functions for both the Binary and Multi-class classification. These functions achieved good attack detection rates than the *ReLU* for 100 epochs in Binary classification. When the same set of experiments were run for 500 epochs the experiments with *ReLU* activation function performed better than the *sigmoid* and *tanh* activation functions. In the case of Multi-class classification, the performance of *ReLU* was good in comparison to the *sigmoid* and *tanh* activation functions. Thus, we decided to set the *ReLU* activation function for the rest of the experiments. All the models were trained using *adam* optimizer with a batch size of 64 for 500 epochs to monitor validation accuracy.

B. FINDING AN OPTIMAL NETWORK TOPOLOGY OF DNN

The following network topologies were used to choose the best network topology for training an IDS model with KDDCup 99.

1) DNN 1 layer
2) DNN 2 layers
3) DNN 3 layers
4) DNN 4 layers
5) DNN 5 layers

For all the above network topologies, we had run 3 trials of experimentation for 300 epochs each. We observed that most of the deep learning architectures learnt the Normal

TABLE 8. Configuration of proposed DNN model.

Layers	Type	Output shape	Number of units	Activation function	Parameters
0-1	fully connected	(None, 1,024)	1,024	ReLU	43,008
1-2	Batch Normalization	(None, 1,024)			4,096
2-3	Dropout (0.01)	(None, 1,024)			0
3-4	fully connected	(None, 768)	768	ReLU	7,87,200
4-5	Batch Normalization	(None, 768)			3,072
5-6	Dropout (0.01)	(None, 768)			0
6-7	fully connected	(None, 512)	512	ReLU	3,93,728
7-8	Batch Normalization	(None, 512)			2,048
8-9	Dropout (0.01)	(None, 512)			0
9-10	fully connected	(None, 256)	256	ReLU	1,31,328
10-11	Batch Normalization	(None, 256)			1,024
11-12	Dropout (0.01)	(None, 256)			0
12-13	fully connected	(None, 128)	128	ReLU	32,896
13-14	Batch Normalization	(None, 128)			512
14-15	Dropout (0.01)	(None, 128)			0
15-16	fully connected	KDDCup 99- NSL-KDD- UNSW-NB15- Kyoto- WSN-DS- CICIDS 2017-	Binary, Multi-class: 1, 5- 1, 5- 1, 10- 1- 1, 5 1, 6	$Sigmoid$ for Binary and $Softmax$ for Multi-class classification	

category patterns of input data for epochs less than 400, while the number of epochs required for discovering Attack category was fluctuating. Both the DNN 1 layer and DNN 2 layers networks have completely failed to learn the attack categories of 'R2L' and 'U2R'. The performance of 'DoS' and 'Probe' attack categories was good with DNN 3 layers in comparison to the DNN 2 layers and DNN 1 layer. The complex network architectures required a large number of epochs in order to reach the optimum accuracy. The performance of DNN 5 layers for various Attack categories and Normal category was good as compared to other DNNs network topologies. By considering all these factors, we've decided to use 5 layer DNNs network for the remaining experimentation process.

In order to increase the speed of training and to avert over fitting, we used batch normalization and dropout (0.01) approach. When we run experiments without dropout, the models ends up in over fitting. Also, the experiments with batch normalization achieved better results in comparison to the networks without batch normalization [10]. For HIDS, the best performed DNNs in NIDS as such used. In HIDS, we had followed hyper parameter tuning methods only in conversion of system calls into numeric representation. For N-gram, the 3 trials of experiments were run with 1-gram, 2-gram, 3-gram and 4-gram with different DNNs network topologies. DNNs network topologies with 3-gram system call representation performed well in comparison to the other N-gram system call representation.

C. PROPOSED DNN ARCHITECTURE

This work proposes a unique DNN architecture for NIDS and HIDS composed of an input layer, 5 hidden layers and an output layer. The hierarchical layers in the DNN facilitate to extract highly complex features and do better pattern recognition capabilities in IDS data. Each layer estimates non-linear features that are passed to the next layer and the last layer in the DNN performs the classification. An input layer contains 41 neurons for KDDCup 99, 41 neurons for NSL-KDD, 43 neurons for UNSW-NB15, 17 neurons for WSN-DS and 77 neurons for CICIDS 2017. An output layer contains 1 neuron for Binary classification for all types of datasets and 5 neurons for Multi-class classification in KDD-Cup 99, 5 neurons for NSL-KDD, 10 neurons for UNSW-NB15, 5 neurons for WSN-DS and 8 neurons for CICIDS 2017. The detailed information and configuration details of the DNN architecture is shown in Table 8. The DNN is trained using the backpropogation mechanism [60]. Generally, the units in input to hidden layer and hidden to output layer are fully connected. The DNN is composed of various components, a brief description of each component is given below.

Fully connected layer: This layer is called as fully connected layer since the units in this layer have connection to every other unit in the succeeding layer. Generally, the fully connected layers map the data into high dimensions. The output will be more accurate, when the dimension of data is more. It uses *ReLU* as the non-linear activation function.

Batch Normalization and Regularization: Dropout (0.01) and Batch Normalization [10] was used in between fully connected layers to obviate overfitting and speedup the DNN model training. A dropout removes neurons with their connections randomly. In our alternative architectures, the DNNs could easily overfit the training data without regularization even when trained on large number samples.

Classification: The last layer is a fully connected layer which uses *sigmoid* activation function for Binary classification and *softmax* activation function for Multi-class classification. The prediction loss function for *sigmoid* is defined using Binary cross entropy and the prediction loss for *softmax* is defined using the Categorical cross entropy as follows:

The prediction loss for Binary classification is estimated using Binary cross entropy given by,

$$loss(pd, ed) = -\frac{1}{N}\sum_{i=1}^{N}[ed_i \log pd_i + (1-ed_i)\log(1-pd_i)] \tag{20}$$

where pd is a vector of predicted probability for all samples in testing dataset, ed is a vector of expected class label, values are either 0 or 1.

The prediction loss for Multi-class classification is estimated using Categorical cross entropy given by,

$$loss(pd, ed) = -\sum_{x} pd(x)\log(ed(x)) \tag{21}$$

where ed is true probability distribution, pd is predicted probability distribution. We have used *adam* as an optimizer to minimize the loss of Binary cross entropy and Categorical cross entropy.

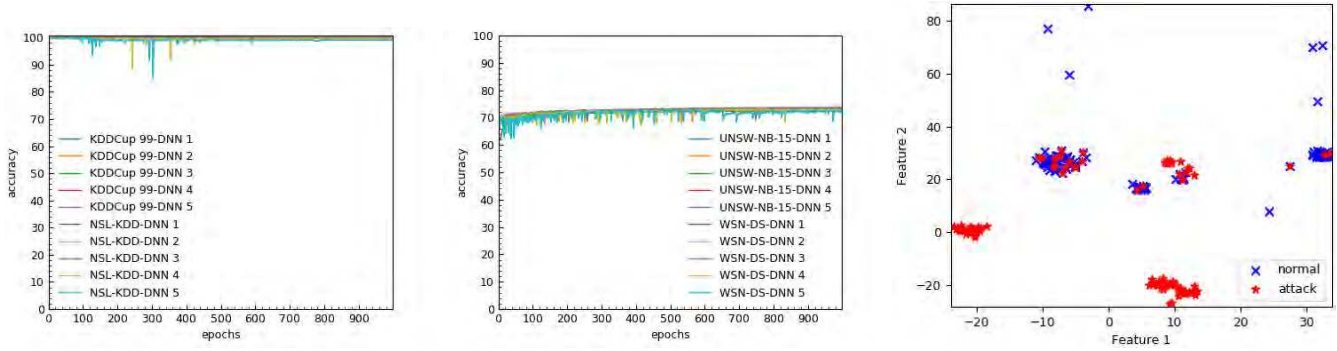
VIII. SCALE-HYBRID-IDS-ALERTNET (SHIA) FRAMEWORK

An IDS has become an indispensable tool for any type of organization or industry due to the wide growing nature of data and internet. There are many attempts that have been made to develop solutions for IDS. These methods used single host for storage and computational resources and the algorithms are not distributed. Using these legacy solutions, it is entirely difficult to monitor and identify intrusions in today's network. This is due to high speed networks and attacks are more and occurring rapidly. The legacy intrusion detection methods which are existing in internet struggle to keep a watch on the networks more efficiently. To address this, based on [61] our work leverages the distributed computing and distributed machine learning models to propose Scale-Hybrid-IDS-AlertNet (SHIA) framework for an effective identification of intrusions and attacks at both the network-level and host-level. The framework provides a scalable design and acts as a distributed monitoring and reporting system. The SHIA enhances the computational performance using the characteristics of distributed computing and distributed machine learning algorithms using a hybrid system placed in different locations in the network. The deployed system is designed to use the computational resource optimally and at the same time with low latency in response while monitoring a critical system. The SHIA framework is basically decomposed into two modules described below:

1) Packet and system call processing module: In this module, the networks which are to be monitored are connected to a single port mirroring switch, which replicates the flow of the entire network traffic of all the switches. The proposed SHIA IDS is required to monitor a network composed of different subnets, each with n number of different machines. The monitored networks are hosts composed of computer machines which allow users to communicate and transfer data. All the traffic generated by the internet were collected, without considering the internal traffic between networks.
   The SHIA framework with the network where one of the switches is used as a port mirroring switch and connected to a traffic collector. This module collects network traffic data using Netmap packet capturing tool and stores it in NoSQL database. Feature vectors are passed into the DNN module, and a copy of data is passed into NoSQL database. Likewise, the system calls and configuration files are collected in a distributed manner by following the methodology provided in [62]. These are passed to NoSQL database and followed by the text representation method to map the system calls to feature vectors. A copy of these feature vectors are dumped into NoSQL database and these feature vectors are again passed into the DNN module for classification.
2) DNN module: From the experimental analysis, we found that the DNN performed well over other algorithms in all cases of HIDS and NIDS. Thus, DNN is used to model the network activities with the aim to detect the attacks more accurately. DNNs require large volume of network and host-level events to learn the behaviors of legitimate and malicious activities. To make the classifier more generalizable, the network traffic activities can be collected in different time with different users in an isolated network. Finally, the DNN module outputs are passed to Front End Broker. This displays the results to the network admin. The implementation details regarding how to conduct a comprehensive experimental analysis will be considered as one of the significant work directions towards future work.

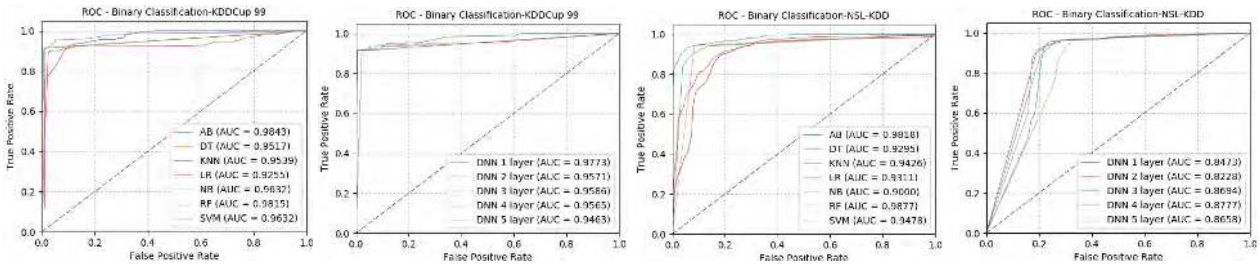
IX. RESULTS

Publicly available NIDS and HIDS datasets were used to evaluate the performance of classical machine learning and DNNs in order to identify a baseline method. These datasets were separated into train and test datasets, and normalized using $L2$ normalization. Train datasets were used to train machine learning model and test datasets were used to evaluate the trained machine learning models. Train accuracy of Multi-class using DNN for KDDCup 99, NSL-KDD and UNSW NB-15, WSN-DS are shown in Figure 4a and 4b



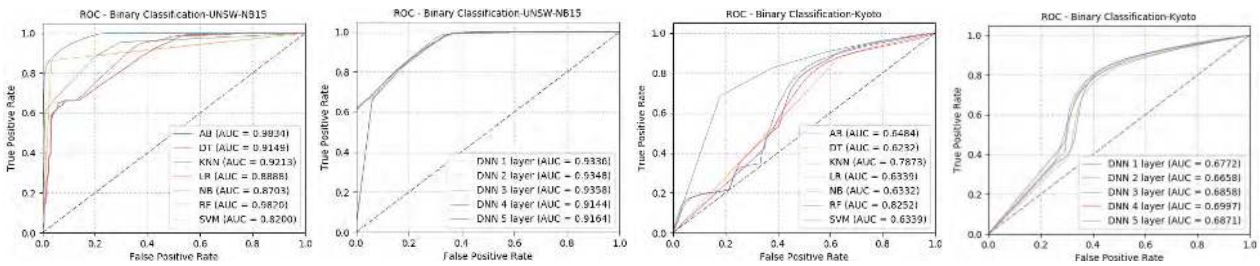
(a) (b) (c)

FIGURE 4. Train accuracy. (a) KDDCup 99 and NSL-KDD. (b) UNSW-NB-15 and WSN-DS. (c) Visualization of 100 connection records with their corresponding activation values of the last hidden layer neurons from Kyoto.



(a) (b) (c) (d)

FIGURE 5. ROC curves of (a) KDDCup 99-using classical machine learning classifiers, (b) KDDCup 99-using DNNs, (c) NSL-KDD-using classical machine learning classifiers, (d) NSL-KDD-using DNNs.



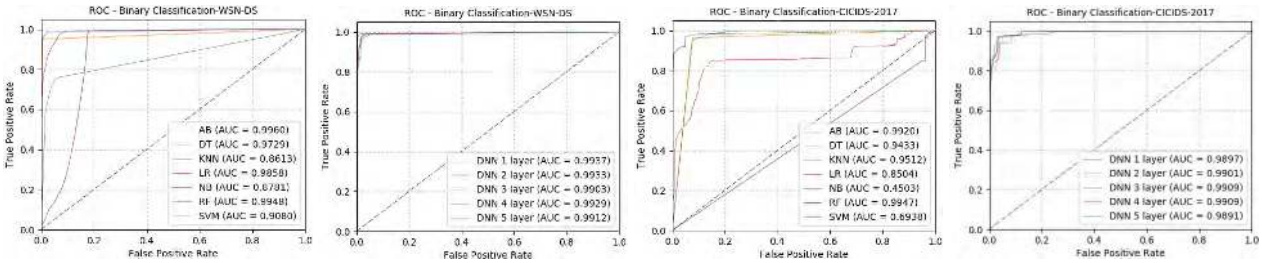
(a) (b) (c) (d)

FIGURE 6. ROC curves of (a) UNSW-NB 15-using classical machine learning classifiers, (b) UNSW-NB 15-using DNNs, (c) Kyoto-using classical machine learning classifiers, (d) Kyoto-using DNNs.

respectively. For KDDCup 99 and NSLKDD datasets, most of the DNN network topologies showed train accuracy in the range 95% to 99%. For UNSW-NB15 and WSN-DS, all the DNN network topologies showed train accuracy in the range 65% to 75%. Several observations were extracted including ROC curve. The ROC curve for KDDCup 99, NSL-KDD, UNSW NB-15, Kyoto, WSN-DS is shown in Figure 5a, Figure 5b, Figure 5c, Figure 5d, Figure 6a, Figure 6b, Figure 6c, Figure 6d, Figure 7a, Figure 7b and Figure 7c, Figure 7d respectively. In most of the cases, DNN performed well in comparison to the classical machine learning classifiers with AUC used as the standard metric. This indicates that the DNN obtained a highest TPR and a lowest FPR and in some cases close to 0. The performance obtained in terms of FPR is less in comparison to other classical machine learning classifiers in all the datasets. The experiments with 3-gram representation performed well as compared to 1-gram and 2-gram.

A. PERFORMANCE COMPARISONS

The detailed results for Binary as well as Multi-class classification of various classical machine learning classifiers and DNNs are reported in Table 9, Table 10 and Table 11, Table 12 respectively. In terms of accuracy noted that the DT, AB and RF classifiers performed better than the other classifiers namely LR, NB, KNN and SVM-rbf. Additionally, the performance of DT, AB and RF classifiers remains the same range across different datasets. However, the performance



(a) (b) (c) (d)

FIGURE 7. ROC curves of (a) WSN-DS-using classical machine learning classifiers, (b) WSN-DS-using DNNs, (c) CICIDS 2017-using classical machine learning classifiers, (d) CICIDS 2017-using DNNs.

TABLE 9. Test results of DNNs for Binary class classification.

Architecture	Accuracy	Precision	Recall	F-score
Binary classification - KDDCup 99				
DNN 1 layer	0.929	0.999	0.914	0.954
DNN 2 layers	0.929	0.998	0.914	0.954
DNN 3 layers	0.930	0.999	0.914	0.955
DNN 4 layers	0.930	0.998	0.915	0.955
DNN 5 layers	0.927	0.994	0.915	0.953
Binary classification - NSL-KDD				
DNN 1 layer	0.801	0.692	0.969	0.807
DNN 2 layers	0.794	0.685	0.965	0.801
DNN 3 layers	0.793	0.684	0.967	0.801
DNN 4 layers	0.794	0.684	0.967	0.802
DNN 5 layers	0.789	0.680	0.963	0.797
Binary classification - UNSW-NB15				
DNN 1 layer	0.784	0.944	0.725	0.820
DNN 2 layers	0.751	0.979	0.649	0.780
DNN 3 layers	0.763	0.963	0.678	0.795
DNN 4 layers	0.765	0.946	0.695	0.801
DNN 5 layers	0.761	0.951	0.684	0.796
Binary classification - WSN-DS				
DNN 1 layer	0.992	0.946	0.971	0.959
DNN 2 layers	0.981	0.842	0.973	0.903
DNN 3 layers	0.970	0.764	0.974	0.856
DNN 4 layers	0.980	0.844	0.967	0.901
DNN 5 layers	0.982	0.931	0.871	0.900
Binary classification - CICIDS 2017				
DNN 1 layer	0.963	0.908	0.973	0.939
DNN 2 layers	0.951	0.897	0.942	0.919
DNN 3 layers	0.944	0.854	0.979	0.912
DNN 4 layers	0.936	0.836	0.976	0.901
DNN 5 layers	0.931	0.827	0.974	0.894
Binary classification - Kyoto				
DNN 1 layer	0.877	0.917	0.950	0.933
DNN 2 layers	0.875	0.915	0.949	0.932
DNN 3 layers	0.877	0.913	0.954	0.933
DNN 4 layers	0.875	0.921	0.943	0.931
DNN 5 layers	0.885	0.913	0.964	0.938

of LR, NB, KNN and SVM-rbf are varied across different datasets. This indicates that the DT, AB and RF classifiers are generalizable and can detect new attacks. While in the

TABLE 10. Test results of DNNs for Multi-class classification.

Architecture	Accuracy	Precision	Recall	F-score
Multi-class classification - KDDCup 99				
DNN 1 layer	0.926	0.938	0.926	0.922
DNN 2 layers	0.926	0.944	0.926	0.920
DNN 3 layers	0.935	0.920	0.935	0.925
DNN 4 layers	0.929	0.911	0.929	0.918
DNN 5 layers	0.925	0.934	0.925	0.921
Multi-class classification - NSL-KDD				
DNN 1 layer	0.778	0.780	0.778	0.760
DNN 2 layers	0.777	0.777	0.777	0.757
DNN 3 layers	0.781	0.785	0.781	0.764
DNN 4 layers	0.780	0.816	0.780	0.763
DNN 5 layers	0.785	0.810	0.785	0.765
Multi-class classification - UNSW-NB15				
DNN 1 layer	0.645	0.614	0.645	0.586
DNN 2 layers	0.660	0.623	0.660	0.596
DNN 3 layers	0.660	0.622	0.660	0.594
DNN 4 layers	0.657	0.606	0.657	0.593
DNN 5 layers	0.651	0.597	0.651	0.585
Multi-class classification - WSN-DS				
DNN 1 layer	0.980	0.983	0.980	0.980
DNN 2 layers	0.969	0.969	0.969	0.968
DNN 3 layers	0.977	0.976	0.977	0.976
DNN 4 layers	0.965	0.964	0.965	0.963
DNN 5 layers	0.964	0.970	0.964	0.966
Multi-class classification - CICIDS 2017				
DNN 1 layer	0.960	0.969	0.960	0.962
DNN 2 layers	0.959	0.970	0.959	0.962
DNN 3 layers	0.962	0.972	0.962	0.965
DNN 4 layers	0.948	0.965	0.948	0.953
DNN 5 layers	0.956	0.962	0.956	0.957

case of multi-class classification the performance of AB is less in compared to DT and RF but performed better than the LR, NB, KNN and SVM-rbf. This is due to the fact that both AB and SVM are not directly applicable for multi-class classification problems. In multi-class, we deal with strengthening the classifier in identifying each individual attacks. Experiments on KDDCup 99 and NSL-KDD, all the classical machine learning classifiers obtained less *TPR* for both 'R2L'

TABLE 11. Test results of various classical machine learning classifiers for binary class classification.

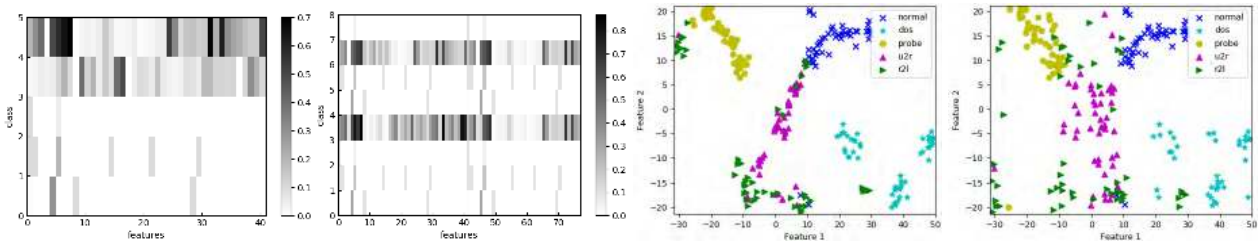
Algorithm	Accuracy	Precision	Recall	F-score
Binary classification - KDDCup 99				
LR	0.811	0.994	0.769	0.867
NB	0.877	0.994	0.852	0.918
KNN	0.925	0.998	0.909	0.952
DT	0.929	0.997	0.915	0.954
AB	0.925	0.996	0.910	0.951
RF	0.927	0.999	0.911	0.953
SVM-rbf	0.877	0.994	0.852	0.918
Binary classification - NSL-KDD				
LR	0.826	0.915	0.744	0.820
NB	0.829	0.865	0.805	0.834
KNN	0.910	0.926	0.905	0.915
DT	0.930	0.928	0.943	0.935
AB	0.934	0.961	0.914	0.937
RF	0.929	0.946	0.919	0.933
SVM-rbf	0.837	0.769	0.993	0.867
Binary classification - UNSW-NB15				
LR	0.743	0.955	0.653	0.775
NB	0.773	0.854	0.805	0.829
KNN	0.810	0.932	0.778	0.848
DT	0.897	0.982	0.864	0.919
AB	0.900	0.985	0.866	0.922
RF	0.903	0.988	0.867	0.924
SVM-rbf	0.653	0.998	0.492	0.659
Binary classification - WSN-DS				
LR	0.970	0.884	0.777	0.827
NB	0.831	0.324	0.765	0.455
KNN	0.943	0.699	0.666	0.682
DT	0.991	0.949	0.951	0.950
AB	0.986	0.897	0.964	0.929
RF	0.996	0.993	0.963	0.978
SVM-rbf	0.915	0.997	0.083	0.153
Binary classification - CICIDS 2017				
LR	0.839	0.685	0.850	0.758
NB	0.313	0.300	0.979	0.459
KNN	0.910	0.781	0.968	0.865
DT	0.935	0.839	0.965	0.898
AB	0.941	0.887	0.918	0.902
RF	0.940	0.849	0.969	0.905
SVM-rbf	0.799	0.992	0.328	0.493
Binary classification - Kyoto				
LR	0.895	0.899	0.995	0.944
NB	0.534	0.922	0.526	0.670
KNN	0.856	0.932	0.905	0.918
DT	0.830	0.925	0.883	0.903
AB	0.889	0.906	0.978	0.940
RF	0.882	0.910	0.963	0.936
SVM-rbf	0.895	0.899	0.995	0.944

and 'U2R' in compared to the other categories such as 'DoS' and 'Probe'. The primary reason is that both the categories of attacks contain very less number of samples in training sets.

TABLE 12. Test results of various classical machine learning classifiers for Multi-class classification.

Algorithm	Accuracy	Precision	Recall	F-score
Multi-class classification - KDDCup 99				
LR	0.801	0.872	0.801	0.804
NB	0.857	0.843	0.857	0.834
KNN	0.921	0.924	0.921	0.912
DT	0.924	0.934	0.924	0.918
AB	0.260	0.821	0.260	0.183
RF	0.925	0.944	0.925	0.918
SVM-rbf	0.895	0.902	0.895	0.890
Multi-class classification - NSL-KDD				
LR	0.612	0.509	0.612	0.530
NB	0.295	0.207	0.295	0.184
KNN	0.731	0.720	0.731	0.684
DT	0.763	0.767	0.763	0.728
AB	0.621	0.651	0.621	0.594
RF	0.753	0.814	0.753	0.715
SVM-rbf	0.702	0.689	0.702	0.656
Multi-class classification - UNSW-NB15				
LR	0.538	0.414	0.538	0.397
NB	0.437	0.579	0.437	0.396
KNN	0.622	0.578	0.622	0.576
DT	0.733	0.721	0.733	0.705
AB	0.608	0.502	0.608	0.526
RF	0.755	0.755	0.755	0.724
SVM-rbf	0.581	0.586	0.581	0.496
Multi-class classification - WSN-DS				
LR	0.944	0.945	0.944	0.943
NB	0.817	0.939	0.817	0.862
KNN	0.926	0.929	0.926	0.926
DT	0.989	0.989	0.989	0.989
AB	0.987	0.987	0.987	0.987
RF	0.994	0.994	0.994	0.994
SVM-rbf	0.915	0.916	0.915	0.880
Multi-class classification - CICIDS 2017				
LR	0.870	0.889	0.870	0.868
NB	0.250	0.767	0.250	0.188
KNN	0.909	0.949	0.909	0.922
DT	0.940	0.965	0.940	0.949
AB	0.641	0.691	0.641	0.653
RF	0.944	0.970	0.944	0.953
SVM-rbf	0.799	0.757	0.799	0.723

Thus during training, the classifiers gives less preference for these attack categories. In terms of accuracy, the performance of the DNN is clearly superior to that of classical machine learning algorithms, often by a large margin in both Binary and Multi-class classification. Moreover, with DNN network topologies, the performance in terms of accuracy is closer to each other's. For Multi-class classification, accuracy, true positive rate (TPR) and false positive rate (FPR) are estimated for each class. The detailed results are shown in Table 15 for KDDCup 99, Table 16 for NSL-KDD, Table 17 for WSN-DS,



(a) (b) (c) (d)

FIGURE 8. Sailency map for a randomly chosen connection record from (a) KDDCup 99, (b) CICIDS 2017 and visualization of 100 connection records with their corresponding activation values of the last hidden layer neurons from (c) KDDCup 99, and (d) NSLKDD.

TABLE 13. Test results using minimal feature sets.

Architecture	Accuracy		
KDDCup 99			
Features	11 features	8 features	4 features
DNN 1 layer	0.842	0.901	0.874
DNN 2 layers	0.898	0.908	0.881
DNN 3 layers	0.908	0.924	0.897
DNN 4 layers	0.924	0.931	0.904
DNN 5 layers	0.921	0.932	0.927
NSL-KDD			
DNN 1 layer	0.621	0.687	0.634
DNN 2 layers	0.637	0.712	0.641
DNN 3 layers	0.689	0.741	0.699
DNN 4 layers	0.684	0.775	0.734
DNN 5 layers	0.752	0.781	0.754

TABLE 14. Test results of host-based IDS.

Method	Accuracy		
ADFA-LD			
N-gram	1-gram	2-gram	3-gram
DNN 1 layer	0.885	0.887	0.894
DNN 2 layers	0.886	0.894	0.897
DNN 3 layers	0.894	0.899	0.904
DNN 4 layers	0.891	0.901	0.912
DNN 5 layers	0.895	0.902	0.917
DNN 5 layers - Keras embedding	0.921		
SVM + tf-idf	0.887		
ADFA-WD			
DNN 1 layer	0.741	0.764	0.774
DNN 2 layers	0.765	0.757	0.785
DNN 3 layers	0.784	0.778	0.794
DNN 4 layers	0.788	0.789	0.801
DNN 5 layers	0.791	0.795	0.812
DNN 5 layers - Keras embedding	0.834		
SVM+tfidf	0.801		

Table 18 for UNSW-NB15 and Table 19 for CICIDS 2017. The proposed method, DNN with 3 layer showed an accuracy of 93.5% that varies -0.32 from the top performed method [3]. However, the proposed method contains less number of parameters. Thus, the proposed method is computationally inexpensive. Moreover, the top performed method [3] uses LSTM, primarily LSTM has been used on raw data [10]. There may be chance that the performance of LSTM can degrade when it sees the real-time datasets or completely unseen samples. The performance obtained in terms of FPR is less comparatively to other classical machine learning classifiers in all the datasets.

False positive occurs when the IDS identifies a connection record as an attack when it is actually a normal traffic. False negative occurs when the IDS fails to interpret a malicious connection record as an attack. In these cases, IDS must be carefully tuned to ensure that these are kept very low. The reported results can be further enhanced by carefully following a hyper parameter selection method with highly complex DNN architecture. In experiments with HIDS, Keras embedding performed better than the N-gram and tf-idf text representation on both HIDS datasets, as shown in Table 14.

Generally, in DNNs, the network connection records are propagated through more than one hidden layers to learn the optimal features. Each hidden layer aims at mapping the data into the higher dimension. Each layer facilitates to understand the significant features towards classifying the network connection into either Normal or Attack and in categorizing the attack into their attack categories. To understand, visualize and analyze the results, the activation values are passed into t-SNE [63]. This converts the high dimensional activation values into low dimensional using Principal Component Analysis (PCA). The low dimensional feature representation for KDDCup 99, NSLKDD and Kyoto is represented in Figure 8c, Figure 8d and Figure 4c respectively. The connection records of 'Normal', 'DoS' and 'Probe' have appeared completely in a different cluster in KDDCup 99. This shows that DNN has learnt the patterns which can distinguish the connection records of 'Normal', 'DoS' and 'Probe'. It has not completely learnt the optimal features to distinguish the connection records of 'U2R' and 'R2L'. This is one of the reasons why few connection records of 'U2R' and 'R2L' have appeared in 'Probe' attack cluster. This shows that the attacks of 'Probe', 'U2R' and 'R2L' have common characteristics. For NSL-KDD, the DNN had same issue as KDDCup 99. For Kyoto dataset, few attacks have appeared in clusters of

TABLE 15. Detailed test results for Multi-class classification- KDDCup 99.

Method	Normal			DoS			Probe			R2L			U2R		
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc
LR	0.978	0.229	0.811	0.798	0.058	0.832	0.001	0.0	0.987	0.014	0.0	1.0	0.063	0.0	0.974
NB	0.517	0.01	0.898	0.99	0.503	0.874	0.0	0.004	0.983	0.514	0.012	0.988	0.001	0.0	0.972
KNN	0.267	0.267	0.642	0.721	0.72	0.617	0.014	0.012	0.975	0.0	0.0	1.0	0.0	0.0	0.972
DT	0.261	0.261	0.646	0.722	0.722	0.617	0.014	0.013	0.974	0.0	0.002	0.998	0.002	0.002	0.971
AB	0.926	0.925	0.24	0.056	0.055	0.266	0.017	0.014	0.973	0.0	0.001	0.999	0.004	0.003	0.969
RF	0.267	0.268	0.642	0.72	0.719	0.616	0.012	0.011	0.976	0.0	0.0	1.0	0.002	0.002	0.971
SVM-rbf	0.298	0.297	0.624	0.693	0.692	0.602	0.009	0.009	0.977	0.0	0.0	1.0	0.0	0.0	0.972
DNN 1 layer	0.994	0.088	0.928	0.939	0.004	0.953	0.732	0.001	0.995	0.243	0.0	1.0	0.155	0.002	0.975
DNN 2 layer	0.995	0.088	0.928	0.942	0.004	0.955	0.764	0.002	0.995	0.243	0.0	1.0	0.089	0.0	0.975
DNN 3 layer	0.981	0.068	0.942	0.962	0.031	0.963	0.706	0.002	0.994	0.0	0.0	1.0	0.0	0.002	0.971
DNN 4 layer	0.946	0.067	0.935	0.961	0.052	0.958	0.765	0.004	0.993	0.0	0.0	1.0	0.0	0.001	0.972
DNN 5 layer	0.991	0.086	0.929	0.939	0.004	0.953	0.752	0.002	0.994	0.043	0.0	1.0	0.136	0.003	0.974

TABLE 16. Detailed test results for Multi-class classification- NSL-KDD.

Method	Normal			DoS			Probe			R2L			U2R		
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc
LR	0.919	0.496	0.682	0.638	0.159	0.77	0.0	0.0	0.899	0.0	0.0	0.997	0.0	0.0	0.879
NB	0.0319	0.085	0.534	0.827	0.861	0.371	0.0	0.008	0.886	0.179	0.069	0.938	0.0	0.016	0.865
KNN	0.977	0.389	0.769	0.731	0.029	0.889	0.590	0.032	0.929	0.119	0.0	0.997	0.0	0.0	0.879
DT	0.975	0.334	0.799	0.788	0.019	0.917	0.605	0.038	0.93	0.419	0.004	0.998	0.077	0.0	0.882
AB	0.631	0.250	0.698	0.861	0.295	0.757	0.336	0.043	0.89	0.438	0.0	0.999	0.168	0.0	0.898
RF	0.976	0.388	0.768	0.758	0.015	0.908	0.648	0.016	0.97	0.522	0.0	0.997	0.049	0.0	0.885
SVM-rbf	0.998	0.507	0.712	0.644	0.0	0.875	0.512	0.0	0.942	0.0	0.0	0.997	0.0	0.0	0.8799
DNN 1 layer	0.973	0.279	0.829	0.777	0.027	0.907	0.61	0.024	0.936	0.433	0.0	0.998	0.241	0.026	0.886
DNN 2 layers	0.971	0.257	0.841	0.764	0.019	0.907	0.703	0.047	0.926	0.224	0.0	0.997	0.199	0.024	0.883
DNN 3 layers	0.973	0.264	0.838	0.772	0.021	0.909	0.636	0.038	0.927	0.239	0.0	0.997	0.26	0.024	0.89
DNN 4 layers	0.974	0.276	0.832	0.764	0.015	0.91	0.663	0.055	0.915	0.672	0.002	0.998	0.242	0.003	0.906
DNN 5 layers	0.974	0.277	0.831	0.795	0.024	0.915	0.634	0.041	0.924	0.269	0.0	0.998	0.229	0.005	0.903

TABLE 17. Detailed test results for Multi-class classification- WSN-DS.

Method	Normal			Blackhole			Grayhole			Flooding			Scheduling		
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc
LR	0.998	0.147	0.934	0.159	0.048	0.844	0.609	0.145	0.807	0.755	0.0	0.988	0.702	0.0	0.973
NB	0.946	0.066	0.946	0.989	0.146	0.87	0.478	0.039	0.875	0.8385	0.017	0.976	0.288	0.0	0.937
KNN	0.992	0.341	0.838	0.485	0.011	0.929	0.387	0.0888	0.809	0.403	0.007	0.969	0.664	0.0	0.96
DT	0.996	0.054	0.97	0.933	0.004	0.98	0.945	0.014	0.981	0.702	0.0	0.986	0.939	0.0	0.996
AB	0.999	0.085	0.964	0.885	0.008	0.977	0.839	0.019	0.957	0.984	0.0	0.999	0.847	0.0	0.98
RF	0.999	0.034	0.98	0.961	0.0	0.991	0.958	0.0	0.986	0.837	0.0	0.998	0.942	0.0	0.997
SVM-rbf	0.999	0.922	0.575	0.0	0.0	0.866	0.142	0.0	0.833	0.014	0.0	0.956	0.075	0.	0.918
DNN 1 layer	0.998	0.027	0.98	0.965	0.078	0.939	0.616	0.007	0.919	0.978	0.005	0.994	0.917	0.0	0.992
DNN 2 layers	0.999	0.091	0.957	0.754	0.073	0.904	0.538	0.046	0.87	0.754	0.0	0.989	0.937	0.0	0.993
DNN 3 layers	0.999	0.107	0.953	0.883	0.037	0.956	0.666	0.026	0.916	0.776	0.0	0.987	0.916	0.0	0.992
DNN 4 layers	0.998	0.145	0.933	0.862	0.071	0.92	0.474	0.033	0.873	0.648	0.0	0.984	0.796	0.0	0.983
DNN 5 layers	0.994	0.047	0.975	0.946	0.069	0.939	0.676	0.019	0.925	0.819	0.0	0.998	0.879	0.0	0.988

normal connection records. This shows that they have similar characteristics and it requires additional features to classify it correctly.

To know the importance of each feature and as well as to identify the significant features, the methodology of [69] has been followed. This uses Taylor expansion for the features of penultimate layer and finds the first order partial derivative of the classification results before placing them through the *softmax* function. This helps to detect the significant features to distinguish the connection record as either

TABLE 18. Detailed test results for Multi-class classification- UNSW-NB15.

Method	Normal			Fuzzers			Analysis			Backdoors		
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc
LR	0.954	0.376	0.729	0.0	0.0	0.999	0.0	0.0	0.995	0.0	0.0	0.941
NB	0.527	0.025	0.837	0.086	0.005	0.997	0.018	0.0	0.9866	0.465	0.276	0.708
KNN	0.936	0.309	0.769	0.0	0.0	0.999	0.035	0.0	0.993	0.349	0.008	0.952
DT	0.967	0.134	0.899	0.279	0.0	0.999	0.447	0.0	0.994	0.713	0.0	0.976
AB	0.992	0.425	0.707	0.0	0.0	0.999	0.0	0.0	0.993	0.0	0.0	0.941
RF	0.984	0.149	0.896	0.146	0.0	0.999	0.555	0.0	0.995	0.718	0.0	0.979
SVM-rbf	0.998	0.532	0.637	0.0	0.0	0.999	0.017	0.0	0.996	0.309	0.0	0.958
DNN 1 layer	0.897	0.289	0.775	0.0	0.0	0.999	0.0	0.0	0.995	0.335	0.0	0.957
DNN 2 layers	0.947	0.299	0.784	0.0	0.0	0.999	0.0	0.0	0.995	0.3355	0.0	0.958
DNN 3 layers	0.956	0.312	0.777	0.0	0.0	0.999	0.076	0.0	0.993	0.335	0.0	0.957
DNN 4 layers	0.915	0.264	0.797	0.0	0.0	0.999	0.0	0.0	0.995	0.345	0.0	0.957
DNN 5 layers	0.928	0.285	0.789	0.0	0.0	0.999	0.0	0.0	0.995	0.344	0.013	0.951
Algorithm	DoS			Exploits			Generic			Reconnaissance		
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc
LR	0.984	0.245	0.806	0.011	0.0	0.897	0.038	0.019	0.807	0.0	0.0	0.928
NB	0.985	0.335	0.737	0.1238	0.013	0.897	0.028	0.0	0.811	0.0	0.0	0.928
KNN	0.977	0.0	0.989	0.088	0.0367	0.878	0.248	0.084	0.7885	0.338	0.053	0.903
DT	0.986	0.048	0.959	0.149	0.024	0.889	0.509	0.029	0.885	0.579	0.088	0.888
AB	0.967	0.0	0.986	0.0	0.0	0.8961	0.269	0.0734	0.805	0.271	0.037	0.914
RF	0.984	0.025	0.983	0.112	0.002	0.9	0.616	0.049	0.887	0.579	0.083	0.897
SVM-rbf	0.926	0.0	0.982	0.0128	0.0	0.895	0.068	0.024	0.806	0.25	0.039	0.916
DNN 1 layer	0.978	0.0	0.999	0.026	0.0	0.895	0.577	0.176	0.776	0.035	0.0	0.926
DNN 2 layers	0.978	0.0	0.997	0.017	0.0	0.898	0.577	0.155	0.797	0.045	0.0	0.925
DNN 3 layers	0.977	0.0	0.993	0.016	0.0	0.894	0.563	0.144	0.803	0.032	0.0	0.929
DNN 4 layers	0.979	0.0	0.992	0.0107	0.0	0.894	0.617	0.180	0.784	0.028	0.0	0.928
DNN 5 layers	0.977	0.0	0.994	0.013	0.0	0.899	0.571	0.166	0.783	0.018	0.008	0.927
Algorithm	Shell code			Worms								
	TPR	FPR	Acc	TPR	FPR	Acc						
LR	0.0	0.0	0.99	0.0	0.0	0.988						
NB	0.0	0.0	0.99	0.0	0.0	0.988						
KNN	0.0	0.0	0.989	0.0115	0.0	0.986						
DT	0.062	0.0	0.99	0.0	0.0	0.988						
AB	0.0	0.0	0.99	0.0	0.0	0.986						
RF	0.039	0.0	0.994	0.0	0.0	0.988						
SVM-rbf	0.0	0.0	0.99	0.0	0.0	0.988						
DNN 1 layer	0.0	0.0	0.99	0.0	0.0	0.988						
DNN 2 layers	0.0	0.0	0.99	0.0	0.0	0.988						
DNN 3 layers	0.0	0.0	0.99	0.0	0.0	0.988						
DNN 4 layers	0.0	0.0	0.99	0.0	0.0	0.988						
DNN 5 layers	0.0	0.0	0.99	0.0	0.0	0.988						

Normal or Attack and categorize an Attack into its categories. The connection record which belongs to 'R2L' is shown in Figure 8a. The features have similar characteristics as of features of 'U2R'. For CICIDS 2017, the connection record which belongs to 'DoS' is shown in Figure 8b. The features have similar characteristics between the 'DoS' and 'DDoS'. This shows that the dataset requires few more additional features to classify the connection record to 'DoS' and 'DDoS' correctly.

Both classical machine learning classifiers and DNNs networks have performed well on KDDCup 99 in comparison to the NSL-KDD. The NSL-KDD is a refined version of KDDCup 99 dataset. Thus, the dataset has unique set of train and test connection records. Moreover, the connection records of NSL-KDD is highly non-linearly separable in comparison to the KDDCup 99. Moreover, the performance of both the classical machine learning classifiers and DNNs considerably less in comparison to the KDDCup 99 and NSL-KDD. On subset of CICIDS 2017, both the classical machine learning classifiers and DNNs performed well. Thus, the proposed SHIA architecture in this work can work well in real-time. The performance of DNNs can be enhanced by carefully following a hyper parameter techniques for NSL-KDD and UNSW-NB 15.

TABLE 19. Detailed test results for Multi-class classification- CICIDS 2017.

Method	Normal			SSH-Patator			FTP-Patator					
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc			
LR	0.905	0.161	0.885	0.499	0.0	0.984	0.488	0.0	0.982			
NB	0.0366	0.0	0.322	0.998	0.25	0.757	0.979	0.317	0.6926			
KNN	0.884	0.033	0.909	0.979	0.029	0.97	0.945	0.0	0.995			
DT	0.928	0.028	0.939	0.941	0.0	0.996	0.972	0.0	0.999			
AB	0.693	0.378	0.678	0.480	0.058	0.927	0.984	0.0	0.965			
RF	0.935	0.035	0.944	0.951	0.0	0.999	0.973	0.0	0.998			
SVMrbf	0.999	0.681	0.798	0.450	0.0	0.988	0.382	0.0	0.979			
DNN 1 layer	0.646	0.645	0.559	0.0	0.018	0.959	0.031	0.040	0.928			
DNN 2 layers	0.644	0.642	0.558	0.0	0.0	0.969	0.038	0.040	0.928			
DNN 3 layers	0.651	0.648	0.561	0.0	0.0	0.963	0.031	0.041	0.927			
DNN 4 layers	0.638	0.639	0.555	0.0227	0.0172	0.958	0.031	0.040	0.928			
DNN 5 layers	0.668	0.666	0.568	0.0156	0.014	0.958	0.031	0.040	0.928			
Algorithm	Web			Bot			DDoS			PortScan		
	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc	TPR	FPR	Acc
LR	0.806	0.0	0.998	0.0	0.0	0.982	0.934	0.0	0.992	0.994	0.078	0.926
NB	0.882	0.019	0.979	0.995	0.012	0.988	0.908		0.985	0.129	0.065	0.879
KNN	0.813	0.0	0.997	0.995	0.040	0.960	0.979	0.0	0.995	1.0	0.0	0.996
DT	0.868	0.0	0.995	0.998	0.038	0.966	0.998	0.0	0.999	1.0	0.0	0.997
AB	0.0	0.0	0.993	0.0	0.100	0.884	0.0	0.0	0.929	0.995	0.04705	0.955
RF	0.841	0.0	0.996	0.998	0.039	0.961	0.995	0.0	0.999	1.0	0.0	0.998
SVM-rbf	0.781	0.0	0.996	0.0	0.0	0.982	0.0	0.0	0.929	0.993	0.0	0.99
DNN 1 layer	0.013	0.0	0.985	0.028	0.021	0.961	0.095	0.088	0.854	0.093	0.092	0.852
DNN 2 layers	0.013	0.011	0.982	0.033	0.032	0.951	0.095	0.088	0.854	0.090	0.090	0.853
DNN 3 layers	0.020	0.007	0.986	0.041	0.034	0.949	0.0951	0.088	0.854	0.090	0.089	0.854
DNN 4 layers	0.013	0.009	0.984	0.033	0.0334	0.950	0.0964	0.088	0.854	0.096	0.088	0.
DNN 5 layers	0.017	0.009	0.988	0.028	0.028	0.959	0.094	0.086	0.855	0.089	0.089	0.855

The detailed test results of ADFA-LD and ADFA-WD datasets are reported in Table 14. N-gram and Keras embedding text representation methods are used to transform system call into numeric vectors with DNNs. For comparative study, tf-idf is used as system call representation method with classical machine learning classifier, SVM. The performance of N-gram and Keras embedding is good in compared to tf-idf. This is due to the fact that both N-gram and Keras embedding have the capability to preserve the sequence information of the system calls. Moreover, the Keras embedding performed well over N-gram representation method. This is due to the fact that it facilitates to capture the relation among system calls. To choose the best value for N in N-gram, the experiments are run with 1, 2 and 3 gram. When the N is increased from 3 to 4, the performance reduced. The experiments with 3-gram representation performed well in compared to 1-gram and 2-gram. However, the N-gram and tf-idf representation produces very large matrix and in some cases this type of matrix can be sparse. Thus there may be chance that using any classifier it is very difficult to achieve best performance on the sparse representation. An additional advantage of Keras embedding is that the weights of the embedding layer is updated during backpropogation.

B. IMPORTANCE OF MINIMAL FEATURE SETS

Feature selection is an important step for intrusion detection. It is an important step in order to identify the various types of attacks more accurately. Without feature selection, there may be a possibility in misclassification of attacks and it would take a large time to train a model [70]. The significance of feature selection method was discussed in detail for intrusion detection using NLS-KDD dataset [71]. They reported that the feature selection method significantly reduces the training and testing time and also showed improved intrusion detection rate. To evaluate the performance of various DNN topologies and static machine learning classifiers, two trials of experiments are run on minimal feature sets on KDDCup 99 and NSL-KDD [3]. The detailed results are reported in Table 13. The experiments with 11 and 8 feature sets performed well in comparison to the experiments with 4 feature set. Moreover, experiments with 11 feature sets performed well in comparison to the 8 feature set. The difference in performance between 11 and 8 minimal feature sets is marginal.

X. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a hybrid intrusion detection alert system using a highly scalable framework on commodity

hardware server which has the capability to analyze the network and host-level activities. The framework employed distributed deep learning model with DNNs for handling and analyzing very large scale data in real-time. The DNN model was chosen by comprehensively evaluating their performance in comparison to classical machine learning classifiers on various benchmark IDS datasets. In addition, we collected host-based and network-based features in real-time and employed the proposed DNN model for detecting attacks and intrusions. In all the cases, we observed that DNNs exceeded in performance when compared to the classical machine learning classifiers. Our proposed architecture is able to perform better than previously implemented classical machine learning classifiers in both HIDS and NIDS. To the best of our knowledge this is the only framework which has the capability to collect network-level and host-level activities in a distributed manner using DNNs to detect attack more accurately.

The performance of the proposed framework can be further enhanced by adding a module for monitoring the DNS and BGP events in the networks. The execution time of the proposed system can be enhanced by adding more nodes to the existing cluster. In addition, the proposed system does not give detailed information on the structure and characteristics of the malware. Overall, the performance can be further improved by training complex DNNs architectures on advanced hardware through distributed approach. Due to extensive computational cost associated with complex DNNs architectures, they were not trained in this research using the benchmark IDS datasets. This will be an important task in an adversarial environment and is considered as one of the significant directions for future work.

ACKNOWLEDGMENT

The authors would like to thank NVIDIA India, for the GPU hardware support to research grant. They would also like to thank Computational Engineering and Networking (CEN) department for encouraging the research.

REFERENCES

[1] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, May 1994.
[2] D. Larson, "Distributed denial of service attacks–holding back the flood," *Netw. Secur.*, vol. 2016, no. 3, pp. 5–7, 2016.
[3] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South Afr. Comput. J.*, vol. 56, no. 1, pp. 136–154, 2015.
[4] S. Venkatraman and M. Alazab, "Use of data visualisation for zero-day Malware detection," *Secur. Commun. Netw.*, vol. 2018, Dec. 2018, Art. no. 1728303. [Online]. Available: https://doi.org/10.1155/2018/1728303
[5] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surveys Tuts.*, to be published. doi: 10.1109/comst.2018.2847722.
[6] A. Azab, M. Alazab, and M. Aiash, "Machine learning based botnet identification traffic," in *Proc. 15th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (Trustcom)*, Tianjin, China, Aug. 2016, pp. 1788–1794.
[7] R. Vinayakumar. (Jan. 2, 2019). *Vinayakumarr/Intrusion-Detection V1 (Version V1)*. [Online]. Available: http://doi.org/10.5281/zenodo.2544036
[8] M. Tang, M. Alazab, Y. Luo, and M. Donlon, "Disclosure of cyber security vulnerabilities: time series modelling," *Int. J. Electron. Secur. Digit. Forensics*, vol. 10, no. 3, pp. 255–275, 2018.
[9] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, 1999. doi: 10.1016/S1389-1286(99)00112-7.
[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[11] Y. Xin *et al.*, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.
[12] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, no. 3, pp. 151–180, 1998.
[13] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proc. IEEE Symp. Secur. Privacy*, May 1996, pp. 120–128.
[14] N. Hubballi, S. Biswas, and S. Nandi, "Sequencegram: n-gram modeling of system calls for program based anomaly detection," in *Proc. 3rd Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2011, pp. 1–10.
[15] N. Hubballi, "Pairgram: Modeling frequency information of lookahead pairs for system call based anomaly detection," in *Proc. 4th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2012, pp. 1–10.
[16] H. Kozushko, "Intrusion detection: Host-based and network-based intrusion detection systems," Independ. Study, New Mexico Inst. Mining Technol., Socorro, NM, USA, 2003.
[17] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, 2000, Art. no. 227261. doi: 10.1145/382912.382914.
[18] A. Ozgur and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ PrePrints*, vol. 4, Apr. 2016, Art. no. e1954.
[19] R. Agarwal and M. V. Joshi, "PNrule: A new framework for learning classifier models in data mining," Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. TR 00-015, 2000.
[20] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets," *Proc. 3rd Annu. Conf. Privacy, Secur. Trust*, 2005, pp. 12–14.
[21] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.
[22] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Gener. Comput. Syst.*, vol. 55, pp. 376–390, Feb. 2016.
[23] M. Alazab *et al.*, "A hybrid wrapper-filter approach for Malware detection," *J. Netw.*, vol. 9, no. 11, pp. 2878–2891, 2014.
[24] W. Hu, W. Hu, and S. Maybank, "AdaBoost-based algorithm for network intrusion detection," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 2, pp. 577–583, Apr. 2008.
[25] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. SIAM Int. Conf. Data Mining*, 2013, pp. 47–58.
[26] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive Bayesian networks in intrusion detection systems," in *Proc. 23rd Workshop Probabilistic Graph. Models Classification, 14th Eur. Conf. Mach. Learn. (ECML) 7th Eur. Conf. Princ. Pract. Knowl. Discovery Databases (PKDD)*, Cavtat-Dubrovnik, Croatia, 2003, p. 11.
[27] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Berlin, Germany: Springer, Oct. 2000, pp. 80–93.
[28] D.-Y. Yeung and C. Chow, "Parzen-window network intrusion detectors," in *Proc. 16th Int. Conf. Pattern Recognit.*, vol. 4, Aug. 2002, pp. 385–388.
[29] W. Li, "Using genetic algorithm for network intrusion detection," in *Proc. United States Dept. Energy Cyber Secur. Group Training Conf.*, 2004, pp. 24–27.
[30] L. Didaci, G. Giacinto, and F. Roli, "Ensemble learning for intrusion detection in computer networks," in *Proc. Workshop Mach. Learn. Methods Appl.*, Siena, Italy, 2002, pp. 1–11.
[31] C. Kolias, G. Kambourakis, and M. Maragoudakis, "Swarm intelligence in intrusion detection: A survey," *Comput. Secur.*, vol. 30, no. 8, pp. 625–642, 2011. doi: 10.1016/j.cose.2011.08.009.
[32] C. yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[33] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BIONETICS)*, 2016, pp. 21–26.

[34] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc.Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2016, pp. 1–5.

[35] F. A. B. H. Ali and Y. Y. Len, "Development of host based intrusion detection system for log files," in *Proc. IEEE Symp. Bus., Eng. Ind. Appl. (ISBEIA)*, Sep. 2011, pp. 281–285.

[36] M. Topallar, M. O. Depren, E. Anarim, and K. Ciliz, "Host-based intrusion detection by monitoring Windows registry accesses," in *Proc. IEEE 12th Signal Process. Commun. Appl. Conf.*, Apr. 2004, pp. 728–731.

[37] E. Aghaei and G. Serpen, "Ensemble classifier for misuse detection using N-gram feature vectors through operating system call traces," *Int. J. Hybrid Intell. Syst.*, vol. 14, no. 3, pp. 141–154, 2017.

[38] B. Borisaniya and D. Patel, "Evaluation of modified vector space representation using ADFA-LD and ADFA-WD datasets," *J. Inf. Secur.*, vol. 6, no. 3, p. 250, 2015.

[39] M. Xie, J. Hu, X. Yu, and E. Chang, "Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2014, pp. 542–549.

[40] B. Subba, S. Biswas, and S. Karmakar, "Host based intrusion detection system using frequency analysis of n-gram terms," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2017, pp. 2006–2011.

[41] W. Haider, G. Creech, Y. Xie, and J. Hu, "Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks," *Future Internet*, vol. 8, no. 3, p. 29, 2016.

[42] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon. (2016). "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems." [Online]. Available: https://arxiv.org/abs/1611.01726

[43] M. Kezunovic, L. Xie, and S. Grijalva, "The role of big data in improving power system operation and protection," in in *Proc. IREP Symp. Bulk Power Syst. Dyn. Control-IX Optim., Secur. Control Emerg. Power Grid (IREP)*, Aug. 2013, pp. 1–9.

[44] M. Tang, M. Alazab, and Y. Luo, "Big data for cybersecurity: Vulnerability disclosure trends and dependencies," *IEEE Trans. Big Data*, to be published. doi: 10.1109/tbdata.2017.2723570.

[45] R. Vinayakumar, P. Poornachandran, and K. P. Soman, "Scalable framework for cyber threat situational awareness based on domain name systems data analysis," in *Big Data in Engineering Applications* (Studies in Big Data), vol. 44, S. Roy, P. Samui, R. Deo, and S. Ntalampiras, Eds. Singapore: Springer, 2018.

[46] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008, Ch. 20, pp. 405–416.

[47] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[48] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 30, no. 1, pp. 1–3.

[49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[50] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection," in *Recent Advances in Intrusion Detection* (Lecture Notes in Computer Science), vol. 2820. Berlin, Germany: Springer, 2003, pp. 220–237.

[51] M. Sabhnani and G. Serpen, "Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set," *Intell. Data Anal.*, vol. 8, no. 4, pp. 403–415, 2004.

[52] Y. Bouzida and F. Cuppens, "Neural networks vs. decision trees for intrusion detection," in *Proc. IEEE/IST Workshop Monitoring, Attack Detection Mitigation (MonAM)*, Sep. 2006, pp. 1–29.

[53] S. T. Brugger and J. Chow, "An assessment of the DARPA IDS evaluation dataset using snort," Dept. Comput. Sci., Univ. California, Davis, Davis, CA, USA, Tech. Rep. CSE-2007-1, 2005.

[54] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, 2000. doi: 10.1145/382912.382923.

[55] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Symp. Comput. Intell. Secur. Defence Appl.*, Jul. 2009, pp. 1–6.

[56] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[57] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. IEEE Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[58] J. Song *et al.*, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," in *Proc. 1st Workshop Building Anal. Datasets Gathering Exper. Returns Secur.*, 2011, pp. 29–36.

[59] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2018, pp. 1–8.

[60] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2101–2104, Sep. 1991.

[61] M. Alazab, "Profiling and classifying the behavior of malicious codes," *J. Syst. Softw.*, vol. 100, pp. 91–102, Feb. 2015.

[62] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 807–819, Apr. 2014.

[63] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[64] J. O. Nehinbe, "A critical evaluation of datasets for investigating IDSs and IPSs researches," in *Proc. IEEE 10th Int. Conf. Cybern. Intell. Syst. (CIS)*, Sep. 2011, pp. 92–97.

[65] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, pp. 21–26, 2015.

[66] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection dataset," in *Proc. Int. Conf. Inf. Sci. Secur. (ICISS)*, Dec. 2016, pp. 1–6.

[67] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

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

[69] K. Simonyan, A. Vedaldi, and A. Zisserman. (2013). "Deep inside convolutional networks: Visualising image classification models and saliency maps." [Online]. Available: https://arxiv.org/abs/1312.6034

[70] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of API call signatures," in *Proc. 9th Australas. Data Mining Conf.*, vol. 121, 2011, pp. 171–182.

[71] A. Alazab, M. Hobbs, J. Abawajy, and M. Alazab, "Using feature selection for intrusion detection system," in *Proc. Int. Symp. Commun. Inf. Technol. (ISCIT)*, Oct. 2012, pp. 296–301.

[72] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for Android Malware detection using various features," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 773–788, Mar. 2019.

[73] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "Madam: Effective and efficient behavior-based android malware detection and prevention," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 1, pp. 83–97, Jan. 2018.

[74] S. Naseer *et al.*, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[75] N. R. Sabar, X. Yi, and A. Song, "A bi-objective hyper-heuristic support vector machines for big data cyber-security," *IEEE Access*, vol. 6, pp. 10421–10431, 2018.

[76] W. Wang *et al.*, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[77] S. Aditham and N. Ranganathan, "A system architecture for the detection of insider attacks in big data systems," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 974–987, Nov. 2018.

[78] M. N. Kurt, Y. Yılmaz, and X. Wang, "Real-time detection of hybrid and stealthy cyber-attacks in smart grid," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 498–513, Feb. 2019.

[79] T. Zhang and Q. Zhu, "Distributed privacy-preserving collaborative intrusion detection systems for VANETs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 1, pp. 148–161, Mar. 2018.

R. VINAYAKUMAR received the B.C.A. degree from the JSS College of Arts, Commerce and Sciences, Mysore, in 2011, and the M.C.A. degree from Amrita Vishwa Vidyapeetham, Mysore, in 2014. He is currently pursuing the Ph.D. degree in computational engineering and networking with the Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India. His Ph.D. thesis is on the application of machine learning (sometimes deep learning) for cyber security and also discusses the importance of natural language processing, image processing, and big data analytics for cyber security. He has published several papers in machine learning applied to cyber security. He has participated in several international shared tasks and organized a shared task on detecting malicious domain names (DMD 2018) as part of SSCC'18 and ICACCI'18.

MAMOUN ALAZAB received the Ph.D. degree in computer science from the School of Science, Information Technology and Engineering, Federation University of Australia. He is currently an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Australia. He is a Cyber-Security Researcher and Practitioner with industry and academic experience. He works closely with government and industry on many projects. He has published more than 100 research papers. He has delivered many invited and keynote speeches, 22 events in 2018 alone. His research interest is multidisciplinary that focuses on cyber security and digital forensics of computer systems, including current and emerging issues in the cyber environment like cyber-physical systems and the Internet of Things with a focus on cybercrime detection and prevention. He convened and chaired more than 50 conferences and workshops. He is an Editor on multiple Editorial Boards, including an Associate Editor of the IEEE Access (2017 Impact Factor 3.5), an Editor of the Security and Communication Networks (2016 Impact Factor: 1.067), and a Book Review Section Editor of the *Journal of Digital Forensics, Security and Law*.

K. P. SOMAN has 25 years of research and teaching experience at the Amrita School of Engineering, Coimbatore. He has around 150 publications in national and international journals and conference proceedings. He has organized a series of workshops and summer schools in advanced signal processing using wavelets, kernel methods for pattern classification, deep learning, and big-data analytics for industry and academia. He has authored the books *Insight Into Wavelets*, *Insight Into Data Mining*, *Support Vector Machines and Other Kernel Methods*, and *Signal and Image Processing—The Sparse Way*, published by Prentice Hall, New Delhi, and Elsevier.

PRABAHARAN POORNACHANDRAN is currently a Professor with Amrita Vishwa Vidyapeetham. He has more than two decades of experience in computer science and security areas. His areas of interests are malware, critical infrastructure security, complex binary analysis, AI, and machine learning.

AMEER AL-NEMRAT is currently a Senior Lecturer (Associate Professor) with the School of Architecture, Computing and Engineering, University of East London (UEL). He is the Leader of the Professional Doctorate in IS and M.Sc. Information Security and Computer Forensics Programs. He is also the Founder and the Director of the Electronic Evidence Laboratory, UEL, where he is involved in cybercrime projects with different U.K. law enforcement agencies. His research interests include security, cybercrime, and digital forensics, where he has been publishing research papers in peer-reviewed conferences and internationally reputed journals.

SITALAKSHMI VENKATRAMAN received the M.Sc. degree in mathematics and the M.Tech. degree in computer science from IIT Madras, Madras, in 1985 and 1987, respectively, the M.Ed. degree from The University of Sheffield, in 2001, and the Ph.D. degree in computer science from the National Institute of Industrial Engineering, in 1993. Her Ph.D. thesis was entitled Efficient Parallel Algorithms for Pattern Recognition.

She has more than 30 years of work experience both in industry and academics and has been developing turnkey projects for IT industry and teaching a variety of IT courses for tertiary institutions in India, Singapore, New Zealand, and Australia, since 2007. She is currently the Discipline Leader and a Senior Lecturer in information technology with Melbourne Polytechnic. She is specialized in applying efficient computing models and data mining techniques for various industry problems and recently in the e-health, e-security, and e-business domains through collaborations with industry and universities in Australia. She has published seven book chapters and more than 130 research papers in internationally well-known refereed journals and conferences. She is a Senior Member of professional societies and editorial boards of international journals and serves as a Program Committee Member of several international conferences every year.