Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777224

MAKING USE OF A DEEP LEARNING SYSTEM FUELED BY BIG DATA FOR INTRUSION DETECTION

[1]Raghunath Kumar Babu D , [2]A. Packialatha
1. Research Scholar Computer Science & Engineering, Vel's Institute of Science, Technology and Advanced Studies, Chennai, Tamilnadu 600117, India, raghunath.d29@gmail.com.
2. Department of Computer Science and Engineering, Vels Institute of Science, Technology, and Advanced Studies (VISTAS), Krishnapuram, Pallavaram, Chennai, Tamil Nadu 600117, India.packialatha.se@velsuniv.ac.in

Abstract:
A machine-learning-based IDS is crucial for protecting our economy and military in light of the increasing global interconnection of internet infrastructures and the enormous amounts of data being produced on a daily basis. For intrusion detection, researchers employ a unified learning model that draws from both classical and modern deep learning techniques. The one-learning-model approach might fail if data distributions and incursion patterns become too complicated. It's possible that a single deep learning model won't be sufficient to properly capture distinguishing patterns for intrusive attacks with little data. With the goal of further enhancing the performance of machine learning-based IDS, we present the Big-Data-based Hierarchical Deep-Learning System (BDHDLS). BDHDLS employs behavioural features and content features to comprehend the intricacies of network traffic and the information contained in the payload. Each deep learning model in BDHDLS is trained using data from a single cluster in mind. As compared to approaches that used a single learning model, this one has the potential to increase the frequency with which hostile incursions are discovered. Because of its parallel training approach and massive data capabilities, BDHDLS significantly reduces the time required to create a model when several machines are employed.

1. Introduction

With the exponential growth in both data production and the degree to which different internet infrastructures throughout the world are linked, security experts find new vulnerabilities in the world's internet infrastructures every month [1]. Due to these security holes, cybercriminals may get access to these systems and carry out their illegal activities [2]. As hackers might corrupt financial, medical, and other sensitive data stored in databases, cyber security experts and designers need to provide a broad variety of intrusion detection systems to safeguard computers and networks [2].

Conventional Intrusion Detection Systems (IDS) rely on very specific descriptions, such rules or signatures, to analyse network traffic and detect intrusions [3]. There is often little chance of a false positive when using a signature-based method. The database holding these rules and signatures must be routinely updated by specialists, and doing so is a time-consuming procedure since new intrusion strategies are devised and produced every day. The evolution of assaults into increasingly complex forms poses a significant challenge when trying to establish acceptable signatures [3].

Scientists have investigated machine learning strategies as an alternative to conventional intrusion detection systems[4].As an alternative to conventional intrusion detection systems, machine learning methods Intrusive and non-intrusive samples are automatically reasoned about so that the best possible detection model parameters may be fitted.

In particular, by analysing the attributes reflected in the network data, sophisticated invasive activity patterns may be uncovered using machine learning techniques [5].

Recent years have seen a reevaluation of the importance of machine learning approaches for intrusion detection [3-6]. This is due in large part to the availability of vast numbers of labeled samples, the rise in computing power, and the advancements in machine learning algorithms. Two of the most common kinds of machine learning used in IDSs are the shallow learning model and the deep learning model [3, 4]. Typically, fewer than three computational layers make up a shallow learning model [3]. Support vector machines, logistical regression, and decision trees are all types of shallow learning models [7]. It is common practice to build just a single shallow learning model for an entire dataset. Finding actionable patterns in a large dataset might be challenging for a single shallow learning model. It has been proven that the shallow learning model is unable to pick up on some mapping functions [7]. For intrusion detection in particular, the shallow learning approach has fallen short since it is unable to make use of the growing wealth of available pattern data.

Deep learning is one of the most significant technological advances in the area of artificial intelligence in the previous decade. Deep learning, as contrast to shallow learning, often requires several layers of brain processing [8, 9]. In the areas of computer vision, voice recognition, automated machine translation, and finance, deep learning has outperformed shallow learning models [10, 11]. Researchers have been motivated to use deep learning to intrusion detection [12-15] due to the successes it has had in other contexts.

Most of the time, just one deep learning model is built using the whole dataset. If there are numerous data points, as there often are in computer vision, a single deep learning model can do an excellent job [10]. When there aren't enough data points to train a single deep learning model, the results may be subpar [10]. When the data distribution of intrusion samples becomes more complex, it becomes more challenging to find a solution to the intrusion detection problem using a single deep learning-based technique. As an instance, due to data limitations, a single deep learning model may be unable to recognise certain patterns in the intrusion attempts.

The first stage is to use big data techniques to extract and priorities certain behavioral and content-related features. Then, using the parallel improved K-means clustering algorithm, the whole dataset is partitioned into independent groups. To a large extent, traffic patterns across samples in the same branch of the tree are consistent [4, 5]. The third step is to perform hierarchical clustering in parallel for each one-level cluster that has a poor quality score. These sub trees are then used to construct cluster hierarchies with numerous levels. That is to say, the whole dataset is split up into different groups at different levels. In the last stage, deep learning models are constructed for each cluster in the tree structure so that they may study its unique data distribution pattern. In the last step, the judgments of the several deep learning models are combined to reach a consensus on whether or not the sample under test is invasive.

As for the remainder of the paper, it's structured as follows. Works cited in the discussion of context are included in Section 2. In Chapter 3, we cover the BDHDLS's five stages. The assessment metrics and methodologies, as well as the training set and testing set, are described in Section 4. The findings and interpretations from the experiments are presented in Section 5. In Section 6, we wrap up the paper and discuss what comes next.

2. Related Work

Due to its low false positive rate[2], traditional signature-based techniques have become the standard in intrusion detection software. The signature-based techniques are able to identify attacks by looking for predefined patterns, such as a predefined set of behavioural sequences in network traffic or a predefined set of known malicious instructions in the packet's payload[2]. Keeping the signature database current is essential since the sophistication of network assaults keeps on rising. These revisions are often time-consuming. It is thus possible that new network threats may go undetected because of the time it takes for an attack technique to be discovered and updated signatures to be released[1, 2]. It's also possible that assaults with significantly altered payloads may go undetected by signature-based approaches [1, 2].

There are three drawbacks to creating a single learning model for a dataset that contains examples of different forms of intrusion. Challenges include (1) scalability, (2) capturing patterns from invasive assaults with few training samples, and (3) learning complex data distributions. Secondly, different forms of intrusion use different tactics and target different kinds of information [2]. New methods of assault are developed annually, contributing to the rapid increase in both the number and complexity of invasive attacks [2]. Combining many intrusive attack types into a single dataset might therefore considerably enhance the overall dataset's complexity and hence raise the difficulty of the learning task [2]. Second, there is a scarcity of data for certain forms of malicious intrusion. It is difficult for the deep learning model to grasp the patterns of these invasive assaults because of the large amount of data in the whole dataset combined with the tiny sample size. Lastly, it's not feasible to examine trends in ever-growing intrusion detection datasets using a single deep learning model you build. Other estimates place the time needed to complete deep learning model training with millions of data at several weeks [22]. Nevertheless, the high cost of training may make it difficult for researchers to rapidly test out new deep learning architectures in order to adapt their detection tactics and methods to constantly evolving attack methods.

Machine learning algorithms have a number of challenges, one of which is picking the right characteristics to use.

Machine learning-based IDS typically uses behavioral characteristics and content features [2]. These two kinds of characteristics are utilized to examine the intrusion pattern from various vantage points. The characteristics of network traffic are the primary emphasis of the behavioral features, and they include source-destination ports/IP addresses, packet-level/flow-level statistics, and the length of time that data is collected. Recent literatures have seen the development of a number of deep neural networks for the purpose of assessing network behavior [23–32]. on the other hand, content characteristics are employed to discover attacks that exploit the information carried by packets.

Exploit techniques, harmful behaviors, or unlawful instructions may be uncovered by a careful analysis of payloads [16]. There are benefits and drawbacks to both content feature based and behavioral feature based methods. Yet, since behavioral features only capture network traffic characteristics, it is possible that these types of intrusions might go undetected by a behavioral feature based solution. In order to identify suspicious behaviors, content feature based techniques are potent resources.

Nevertheless, polymorphism and metamorphism may be used to obfuscate payloads, which can undermine content feature based techniques. However, the adversary has a hard time concealing its actions[2]. Most recent studies make use of either behavioural or content aspects. As far as we can determine, there is a lack of comprehensive and insightful study that combines behavioural and content elements.

A further significant problem is scalability in content feature extraction. Payloads may include billions of potential content attributes, making big data approaches ideal for processing such a massive quantity of information [30, 33].

Hence, all deep learning models in the tree collaborate to reach a conclusion, with each model tailored to investigate the data distribution for a distinct class of invasive assaults. For each family of intrusive attacks, BDHDLS has a greater likelihood of learning a unique data distribution pattern than earlier intrusion detection methods. In order to better comprehend the ever-increasingly complicated distribution of intrusion data, BDHDLS takes into account both behavioural characteristics and content information, as opposed to prior techniques that simply used behavioural features or content features. To solve the scalability problems of IDS, BDHDLS uses large data methods to extract features and parallel approaches to complete clustering and model training.

The multi-tiered deep learning approach for malware detection [34] inspired the present study. This study for intrusion detection presents novel difficulties and calls for distinctive approaches to choose relevant characteristics and construct multi-level clusters as compared to malware detection. In particular, the feature selection and clustering processes need the use of big data approaches to accelerate the time- and resource-intensive computing process. In this study, feature extraction from network traffic's behavioral and content characteristics is crucial to making the right choice of deep learning architecture and its hyper parameter. So, the deep learning training and parameter tweaking in this study is quite different from the model development method for malware detection.

3 BDHDLS for Intrusion Detection

The building of BDHDLS consists of five stages:

At the first stage, we use big data approaches to generate behavioral traits and content features;

In the second stage, we will use Spark's parallel enhanced K-means method to divide the dataset into many smaller clusters. Next, we'll parallelize the generation of multi-level cluster trees (stage 3); constructing a deep learning model for each cluster, the fourth phase;

The fifth stage involves combining the results of many deep learning models working on separate clusters to determine if a given sample is harmful or not. Fig. 1 is a flowchart depicting the steps required to create a BDHDLS.

3.1 The creation of both behavioral and content characteristics

A machine learning model's performance is very sensitive to the features it uses, making feature creation and extraction a crucial process.

Producing behavioral and content characteristics is explored here.

At the beginning, behavioral traits are created by monitoring network traffic. This work's behavioral characteristics consist of the total bytes transferred, a variety of packets-level and flow-level statistics, the protocol type, and the length of time data was collected [16]. To rephrase, we employ behavioral traits to mirror elements of network traffic.

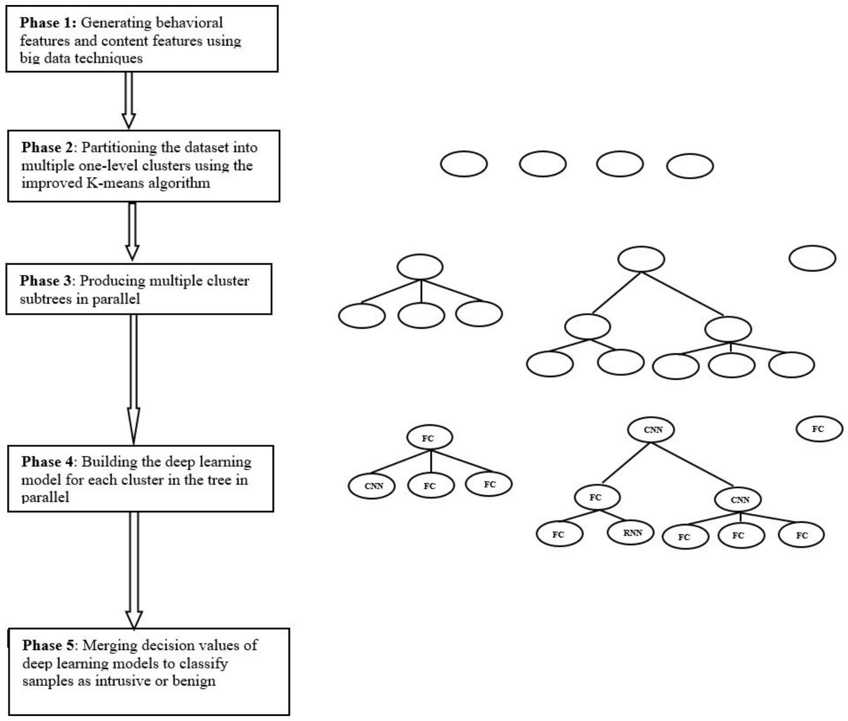
Phase 1: Generating behavioral features and content features using big data techniques
Phase 2: Partitioning the dataset into multiple one-level clusters using the improved K-means algorithm
Phase 3: Producing multiple cluster subtrees in parallel
Phase 4: Building the deep learning model for each cluster in the tree in parallel
Phase 5: Merging decision values of deep learning models to classify samples as intrusive or benign
FC
CNN
FC
FC
CNN
FC
FC
CNN
FC
RNN
FC
FC
FC

Fig. 1 The FC in this diagram indicates a Full Connected feed forward neural network, which is used in the construction of BDHDLS.

After the necessary behavioral characteristics have been generated, the payload's content features may be retrieved to disclose the critical information that had previously been hidden [16]. Attackers rely on payloads as the primary starting point for executing a sequence of malicious commands. Sliding a window of size n bytes produces n-grams (n-byte sequences), the output of the first step of content feature extraction. Frequencies of unique n-grams appearing in the benign samples and the invasive samples are then calculated once the payloads for all samples in the full dataset have been analyzed. It to employ every byte n-gram characteristics retrieved from payloads is typically not a good idea. Since most of these binary n-gram characteristics are likely to be noisy, redundant, or irrelevant, training a classifier using them might drastically degrade the classifier's efficacy. These issues may be sidestepped by first sorting candidate binary n-gram features according to predetermined criteria, which will allow for the selection of a limited subset of features with the highest discriminating power during model formation.

In the literature, information gain has been shown to be one of the most useful metrics for selecting features[2, 35], we use it as the criterion for selecting which content qualities to concentrate on in this research. In this research, we utilise information gain to measure how well a feature classifies the training data. As a metric, the information gain measures how much the entropy of the training data is expected to reduce as a result of a feature-based partition.

For reliable sample categorization, the most informative attribute should be used. The information gain is defined in Eq. (1)[35] as follows.

$$\text{Gain}(A)=\text{Info}(D)-\text{Info}_A(D) \qquad (1)$$

where Info.D/ is the primary data need, and InfoA.D/ is the modified need after separating on dimension A. The formula for Info.D/ is (2):

$$\text{Info}(D) = -\frac{\text{pos}}{\text{total}}\log_2\frac{\text{pos}}{\text{total}} - \frac{\text{neg}}{\text{total}}\log_2\frac{\text{neg}}{\text{total}} \qquad (2)$$

if I is the whole set of samples, then pos is the number of intrusive samples, total is the full set, and neg is the number of benign samples, then total + neg = pos. I in Eq. represents the whole set of data (2). InfoA.D/ is calculated as follows:

$$\text{Info}_A(D) = \sum_{v\in(0,1)} \frac{\text{total}_v}{\text{total}}\left(-\frac{\text{pos}_v}{\text{total}_v}\log_2\frac{\text{pos}_v}{\text{total}_v} - \frac{\text{neg}_v}{\text{total}_v}\log_2\frac{\text{neg}_v}{\text{total}_v}\right) \qquad (3)$$

Parallel n-gram extraction from m cloud nodes is used for each training file (Fig. 2, Step 1). One positive (intrusive) sample and one negative (benign) sample both include the byte sequence 2-gram EE12 from Node 1. The first step in Fig. 2 shows that the equivalent notations for Nodes 1 and 2 are (EE12, C) and (EE12,), respectively. Step 2 of Fig. 2 demonstrates how the in-memory Map Reduce programme first gathers labels of identical n-grams. The resulting paired EE12s are (EE12, C;).

Reducers on the cloud cluster's m nodes get the compiled n-grams. Each reducer counts the accumulated sum to determine how many "yes" and "no" labels there are. If there are more samples in the intrusive set that show the characteristic of interest, then the count is positive .The amount of negative counts in this situation represents the frequency with which this property arises in samples that are otherwise harmless. Figure 2's second step is a visual representation of this idea (EE12, 1, 1).Step 3 of Fig. 2 involves computing the information gain of each byte n-gram feature using an in-memory Map Reduce programme, and then sorting the features based on their information gain. In the fourth step of Fig. 2, the reduction phase selects the optimal feature by maximising the information gain of a single node. Currently, the map phase has no effect. Each component of a content feature set may be seen as a binary feature. For each content feature, its value is set to 1 if that feature can be found in the sample data, and 0 otherwise.Major procedures to extract key content elements using the Spark framework are shown in Figure 2. Spark ensures that m of the cluster's nodes get an equal number of training dataset samples.

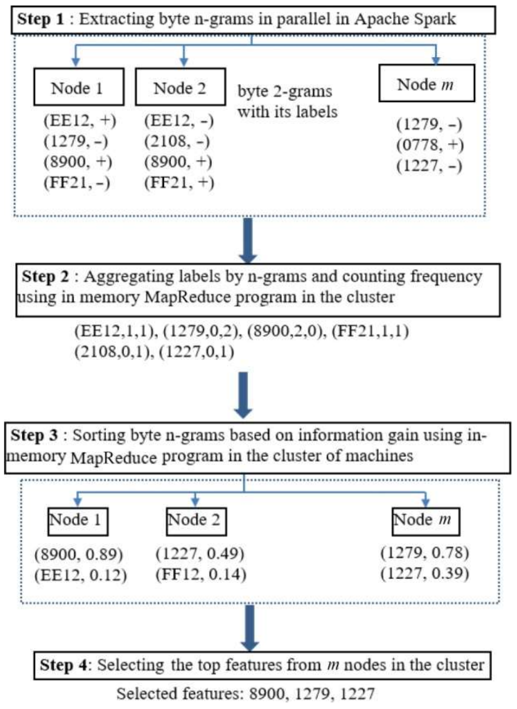
Step 1 : Extracting byte n-grams in parallel in Apache Spark
Node 1
Node 2
Node m
byte 2-grams with its labels
(EE12, +) (1279, -) (8900, +) (FF21, -)
(EE12, -) (2108, -) (8900, +) (FF21, +)
(1279, -) (0778, +) (1227, -)
Step 2 : Aggregating labels by n-grams and counting frequency using in memory MapReduce program in the cluster
(EE12,1,1), (1279,0,2), (8900,2,0), (FF21,1,1) (2108,0,1), (1227,0,1)
Step 3 : Sorting byte n-grams based on information gain using in-memory MapReduce program in the cluster of machines
(8900, 0.89) (EE12, 0.12)
(1227, 0.49) (FF12, 0.14)
(1279, 0.78) (1227, 0.39)
Step 4: Selecting the top features from m nodes in the cluster
Selected features: 8900, 1279, 1227

Fig. 2 Important aspects from material may be extracted in a few major phases with the help of the Spark framework.

Parallel n-gram extraction from m cloud nodes is used for each training file (Fig. 2, Step 1). One positive (intrusive) sample and one negative (benign) sample both include the byte sequence 2-gram EE12 from Node 1. The first step in Fig. 2 shows that the equivalent notations for Nodes 1 and 2 are (EE12, C) and (EE12,), respectively. Step 2 of Fig. 2 demonstrates how the in-memory Map Reduce programme first gathers labels of identical n-grams. The resulting paired EE12s are (EE12, C;).

Reducers on the cloud cluster's m nodes get the compiled n-grams. Each reducer counts the accumulated sum to determine how many "yes" and "no" labels there are. If there are more samples in the intrusive set that show the characteristic of interest, then the count is positive. The frequency with which this attribute appears in samples that are otherwise unharmful is represented by the number of negative counts in this scenario. Step 2 of Fig. 2 illustrates this concept (EE12, 1, 1). Step 3 of Fig. 2 involves computing the information gain of each byte n-gram feature using an in-memory Map Reduce programme, and then sorting the features based on their information gain. In the fourth step of Fig. 2, the reduction phase selects the optimal feature by maximising the information gain of a single node. Currently, the map phase has no effect. Each component of a content feature set may be seen as a binary feature. For each content feature, its value is set to 1 if that feature can be found in the sample data, and 0 otherwise.

3.2 Clustering at the atomic-level

K-means is used to create one-level clusters based on the produced behavioural and content attributes. Standard Kmeans algorithms have serious problems with scalability and initialization. The starting points for clusters in the classic K-means algorithm[36] are chosen

at random from the data set. This method may provide suboptimal partitions that vary from global optimality.
For instance, because to the random selection of initial cluster centres, just a few clusters may collect the majority of the data, while the rest clusters may collect only a tiny number of samples. Classic K-means algorithms suffer the same problem of successfully processing a large number of samples[36].In this study, the parallel enhanced K-means clustering utilising Apache Spark[37] is created to address the scalability and random selection concerns.
To counteract the drawbacks of random initialization, the parallel K-means method makes use of greedy initialization techniques[38]. The greedy initialization method seeks to choose appropriate starting points such that the resulting clustering solution more faithfully reflects the underlying data distribution.
Parallel K-means clustering consists of two phases: the mapping phase and the reduction phase, which occur in each iteration. When mapping is happening, the nearest centroid of a sample is calculated no matter where it is placed inside a data partition. Each section of the dataset is then distributed to different computers. After collecting the centroid partial sum from each data partition for one cluster, the centroid is recalculated for each cluster during the reduction process. As a result, the clustering approach can easily process millions of samples. Calculating the distance between a sample and the centre of a cluster uses the following formula: (4).

$$\mathrm{dist}(x, c) = \sum_{i=1}^{N} |F_x(i) - F_c(i)| \qquad (4)$$

N is the number of features; Fx.i is the value of feature I for sample x; Fc is the value of feature I at the clust's centroid.

3.3 Generation of hierarchical based cluster tree

The purpose of the enhanced K-means clustering's one-level clusters is to place samples with similar patterns together. Several of these one-level clusters have poor quality, as shown by the analysis of one-level clustering. Below is an equation that describes the cluster's quality:

$$\mathrm{Cluster\_Quality}(\%) = \max(P_{\mathrm{benign}}, P_{\mathrm{intrusive}}) \qquad (5)$$

In this group, the Benign fraction represents the proportion of harmless samples, while the Intrusive fraction represents the proportion of hazardous ones. For instance, we may claim that the quality of a cluster is 80% if there are 20% benign samples and 80% invasive samples in the cluster.
The low-quality one-level clusters are sorted into high-quality subclusters using parallel multi-level clustering. There are two phases to parallel multi-level clustering: (1) local discovery and (2) merging. Each low-quality one-level cluster is subjected to agglomerative hierarchical clustering[35] during the local discovery phase. Unless the quality of the combined clusters falls below the specified threshold, the subclusters inside these one-level clusters will keep merging. After the completion of the local discovery phase, a cluster subtree forest is constructed.
In order to merge two trees, their root clusters are joined until their overall quality drops below a threshold value. In this research, we use a hierarchical tree to drill down into the granular subset of the sample data and uncover its underlying distribution patterns.

3.4 Cluster-specific deep learning model training

By expanding on clusters established at a lower level, the hierarchical clustering approach has the potential to unearth additional high-quality clusters. With a multi-level hierarchical tree structure, the noisy and irrelevant information may be integrated in each cluster, making it difficult to accurately compute the distance function for identifying similarity across samples. The efficiency of an intrusion detection system might be severely hampered by the presence of such noisy and irrelevant data.

Several layers of feed forwardly coupled compute units make up the deep FC. When it comes to deep learning models for determining transformation functions, FC is the most general. FC may struggle to deal with data that has unusual qualities since its design does not account for its structure.

There is a subset of neural networks known as RNNs. RNNs are able to provide results that are dependent on those that came before them because of the way they handle sequential data. RNNs are designed to look at the order of payloads inside a single sample to pick up on the sequential dependence[17]. The deep RNN architecture is shown in Figure 4.

The deep learning model is built to examine not just the typical behaviour of each cluster's traffic but also the unique information included inside each cluster's traffic payload. Each deep learning model may zero down on a subset of samples that are most relevant to the task at hand.

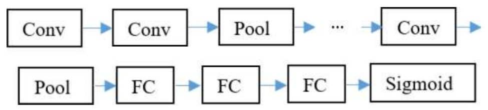

Fig. 3 Diagram for CNN.

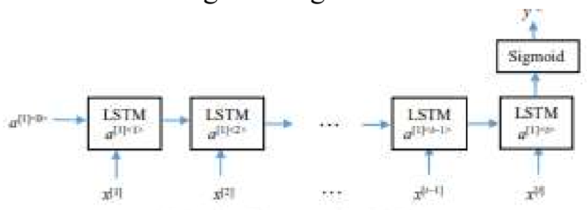

Fig. 4 Diagram for RNN.

without being diverted by irrelevant information. This means that the efficiency of each deep learning model's training may be improved.

A deep FC, CNN, or RNN is built around a function translating the input vector to the output vector over several layers. Eq. (6)[39] describes the mapping function for each layer in a deep learning model.

$$a_{i+1} = f_i(W_i \times a_i + b_i) \quad (6)$$

where ai is the i-th layer's activation, Wi is the i-th layer's weight matrix, bi is the i-th layer's bias, and fi is the activation function. The loss function is a statistical tool for comparing observed and expected values. Equation (7.3)[39] defines the loss function for all three neural network classes:

$$\mathrm{Loss}(y, \hat{y}) = -\sum_{i=1}^{M}[y_i \log \hat{y}_i + (1 - y_i) \log \hat{y}_i] \quad (7)$$

where the real label for sample I is yi, where 0 represents a benign sample and 1 signifies malware. For each sample I, where I is an integer between 1 and M, our deep learning model produces an output represented by yOi.

While optimising deep learning model parameters, the cost function, which is generated from the loss function, provides invaluable information. The cost function is shown to be defined by

$$J(w_1, b_1, \ldots) = \frac{1}{m} \sum_{i=1}^{m} \text{Loss}(y^{(i)}, \hat{y}^{(i)}) \qquad (8)$$

Eq. (8).

To optimise for all weights and biases, we have an optimization cost given by J.w1; b1.

All of the network's settings are fine-tuned within a single round of gradient descent by using the formula represented by Eq (9).

$$w = w - \alpha \frac{\partial \, \text{Loss}(y, \hat{y})}{\partial w} \qquad (9)$$

3.5 Decision fusion algorithm

As shown in Eq. (1), the decision function for cluster k to label sample x as benign or invasive in the deep learning model is as follows.

$$f_k(x) = \text{Sigmoid}(W_k \times y_k + b_k) \qquad (10)$$

where the activation of the final hidden layer in cluster k of the neural network is denoted by yk, and Wk is the weight matrix that links the hidden layer in cluster k to the output layer. For each sample, the decision value indicates how confident the deep learning model was in its classification of the sample as non-invasive or invasive.

The availability of training data for deep learning models is very variable between the many organizations that construct these models. The z-score[35] is used to normalize the deep learning model's classification value, fk.x/, so that it may be fairly compared to the results of other models. We can get the normalized decision value of cluster k for sample x:

$$\text{decision\_value}_k(x) = \frac{f_k(x) - \text{mean}_k}{\delta_k} \qquad (11)$$

1) where meank represents the average classification values of the deep learning model for cluster k and k represents the standard deviation of those values. An increased decision value indicates that the deep learning model is more certain of its classification of the training sample.

$$\text{dist}(k, x) = \sum_{i=1}^{N} |F_x(i) - F_k(i)| \qquad (12)$$

Values of features at index I for sample x (Fx.i /) and the centroid of cluster k (Fk.i /) are used in Equation (12). Eq. defines the logistic function that will be used to smooth the distance between the sample x and the cluster k.

$$\text{smooth\_dist}(k, x) = \frac{1}{1 + e^{-\text{dist}(k,x)}} \qquad (13)$$

(13) in where k represents cluster k and x represents the provided sample. Hence, we can define the weighted decision value for a sample x as Eq.

$$\psi(k, x) = \text{decision\_value}_k(x) \times \text{smooth\_dist}(k, x)$$

(14) If the deep learning models for a specific cluster have a training accuracy below the specified threshold, they will be disregarded in the decision-making process. The samples are then classified based on the greatest weighted decision value. The comprehensive B-building method is shown in DLS.

4. Datasets and Experimental

To begin, we'll go through the independent test datasets that will be used in conjunction with the 5 2 CV F test, and then we'll get into the specifics of the performance measures and model settings.

4.1 Collections of data for the F test of independence and the 5 2 cross validation

While developing the IDS model, it is necessary to take into account both a complete set of intrusions and a profile of the normal, ongoing traffic in the network[16]. To complete the dataset, the raw data must also be included, which contains information about the payload. The raw data, including the payload information, is essential for a clear and full view of how invasive assaults may influence the network and how the servers may react to such an incursion[16].NSL-KDD[40] and Kyoto2009[41] are two examples of popular public datasets, however none of them contains any raw traffic data. Raw traffic data labelled as benign and intrusive samples is provided in just three publicly accessible benchmark datasets (DARPA1998[42], ISCX2012[16], and CICIDS2017[43]). A single network flow, which consists of several network packets exchanged between two parties, is considered a "sample" in this study.

These bytes are collected from a number of different packets of communication and used to create a single sample.

Malicious samples and four kinds of invasive samples (DoS, Probe, U2R, and R2L) make up the DARPA1998 dataset[42]. 3.5 million samples may be found in the DARPA1998 dataset. The ISCX2012 dataset is a publicly available benchmark dataset that records every interaction and payload sent across a certain network. There are several multi-stage assaults in it, and the ambient noise and movement are spot-on[16]. With 1.4 million non-harmful samples and 41,000 potentially harmful samples, the ISCX2012 dataset is very comprehensive. The ISCX2012 dataset is more realistic than the DARPA1998 dataset[16] because it includes four kinds of intrusive samples: BFSSH, DDoS, HTTPDoS, and Infiltrating. The most up-to-date dataset including both safe and harmful samples is the CICIDS2017 dataset. The CICIDS2017 collection contains intrusive samples from a variety of sources, such as Botnet, Web Attack XSS, Web Attack BF, Patator, Port Scan, and Dos[43]. There are 2,800,000 data points in the CICIDS2017 dataset.

Many computational models are evaluated in this paper using these three datasets. During model training and the combined 5 2 cross validation F test, 80% of samples are chosen at random from each dataset. The remaining 20% are used as a separate testing group.

4.2 Performance evaluation metrics

Comparing the intrusion detection system's true positive rate, false positive rate, and accuracy to established norms is one way to evaluate its performance. In Eq. (15)[45], TPR is defined as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad (15)$$

where True Positive (TP) is the amount of intruder samples correctly recognised and False Negative (FN) is the amount of intruder samples incorrectly classified as safe. This is the detection rate, or TPR, which is related to the TPR. As an equation (Eq), FPR defines the relationship between two variables (16) [45]

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \qquad (16)$$

where True Negative (TN) is the number of benign samples that were correctly recognised, and False Positive (FP) is the number of benign samples that were incorrectly labelled as invasive. Another name for FPR is the false alarm rate. Accuracy is defined as in Eq. (17):

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \qquad (17)$$

5 .Experimental Result and Analysis

The independent test and the combined 5 2 CV F test are provided here, together with experimental findings for three datasets.

5.1 Evaluation of Several Feature Sets for Efficiency

First, we evaluate the performance of various feature sets. TPR and accuracy (ACC) are compared across many feature sets in the ISCX2012 dataset in Figure 6.

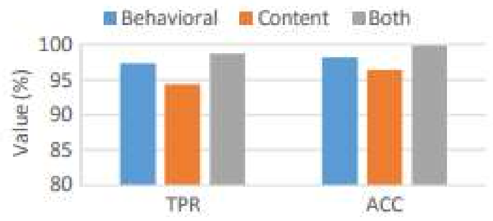

Fig. 6 In the ISCX2012 dataset, TPR and ACC were calculated for a variety of feature sets.

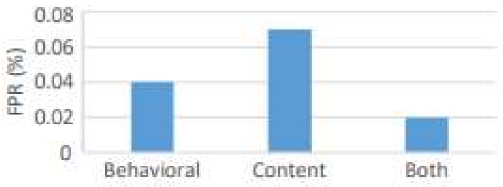

Fig. 7 FPR for the ISCX2012 dataset's various feature sets.

5.2 Results for ISCX2012 dataset

Here, we take a look at the outcomes of third-party evaluations conducted on the ISCX2012 data set. The ISCX2012 dataset is next subjected to a combined 5 2 CV F test, followed by statistical analysis.

5.2.1 Evaluations conducted on the ISCX2012 dataset independently

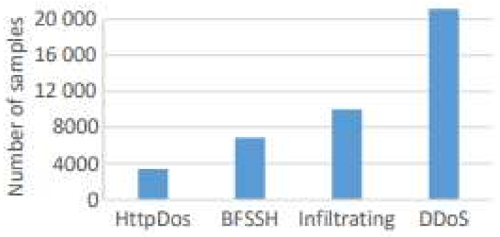

Fig. 8 The total number of instances of various intrusions in the ISCX2012 dataset.

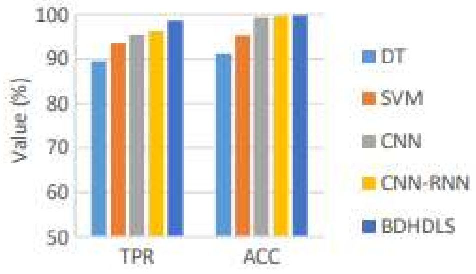

Fig. 9 With the ISCX2012 dataset, we provide TPR and ACC.

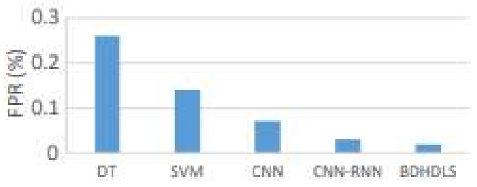

Fig. 10 FPR for ISCX2012 dataset.

Sample counts for several invasive assaults in the ISCX2012 dataset are shown in Figure 8. The ISCX2012 dataset's TPR and ACC are shown in Fig. 9. A comparison of the false positive rates (false alarm rates) of these five models is shown in Fig. 10. The HAST-I model is used for CNN implementation, whereas the HAST-II model is used for CNN-RNN implementation[17].

BDHDLS boosts TPR by 2.5% when compared to the CNN-RNN model. FPR of It has been found that BDHDLS is similar to other computational models.

The results of the deep learning models are shown to be superior to those of the shallow learning models in both figures 9 and 10.

As compared to other single deep learning systems, BDHDLS fares better since it makes use of a hierarchical learning framework.

5.2 Results for CICIDS2017 dataset

The outcomes of external evaluations performed on the CICIDS2017 dataset are detailed here. Finally, we do the statistical test for the combined 5 2 CV F of the CICIDS2017 dataset is conducted.

Model	FPR	TPR	ACC
DT(1)	<0.01	<0.01	<0.01
SVM(2)	<0.01	<0.01	<0.01
CNN(3)	<0.01	0.02	0.07
RNN-CNN(4)	0.08	0.03	0.09
BDHDLS(5)	N/A	N/A	N/A

Table 1 "*p* value by F test" for binary classification in theISCX2012 dataset.

5.2.1 Evaluations conducted on the CICIDS2017 dataset independently

As seen in Figure 11, the CICIDS2017 dataset contains a large number of examples representing a variety of invasive assaults. These are the results from an external testing dataset. As shown in Fig. 12, TPR and ACC for autonomous testing on the CICIDS2017 dataset are provided. Figure 13 compares the CICIDS2017 false positive rate (false alarm rate) across five different models. As compared to CNN-RNN, the TPR is improved by around 2 percentage points because to the BDHDLS.

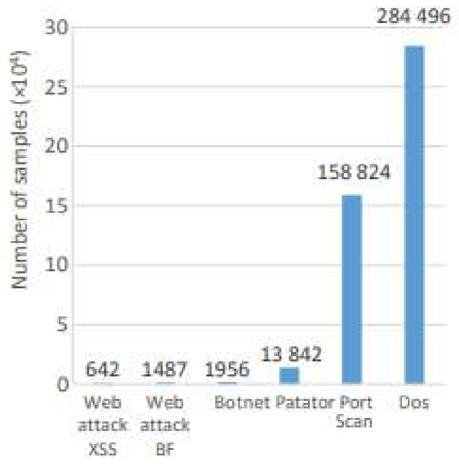

Fig. 11 In the CICIDS2017 dataset, this is the total number of samples for the various intrusion types.

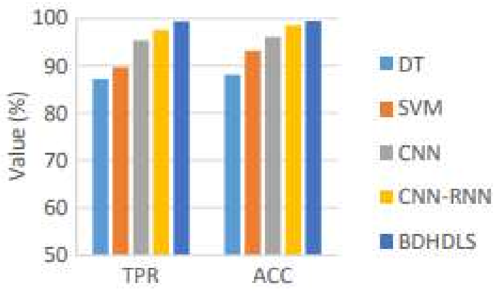

Fig.12 TPR and ACC for CICIDS2017 dataset

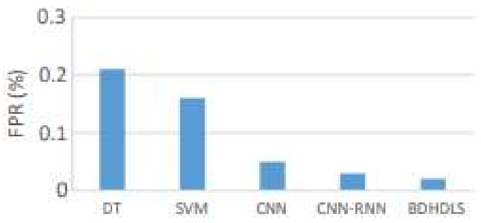

Fig. 13 FPR for CICIDS2017 dataset.

5.3 DARPA1998 Dataset Outcomes

We provide an independent analysis of the DARPA1998 dataset's test data. Intruder sample counts by kind are shown in Figure 14 for the DARPA1998 dataset. Little information is available for travel in the U2R and R2L directions. The results of a third-party test dataset are shown below. The results of four distinct algorithmic models for binary classification on the DARPA1998 dataset are compared and contrasted in Figures 15 and 16.

Model	FPR	TPR	ACC
DT(1)	<0.01	<0.01	<0.01
SVM(2)	<0.01	<0.01	<0.01
CNN(3)	<0.01	<0.1	0.04
RNN-CNN(4)	0.09	0.06	0.08
BDHDLS(5)	N/A	N/A	N/A

Table 2 "*p* value by F test" for binary classification in the CICIDS2017 dataset.

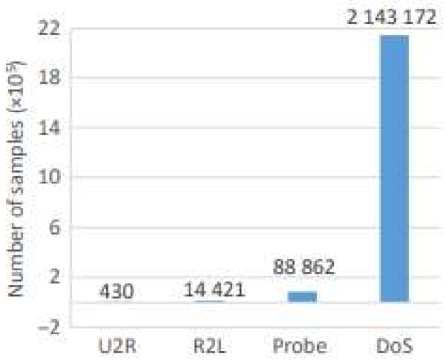

Fig. 14 Number of samples for different intrusive attacks inthe DARPA1998 dataset.

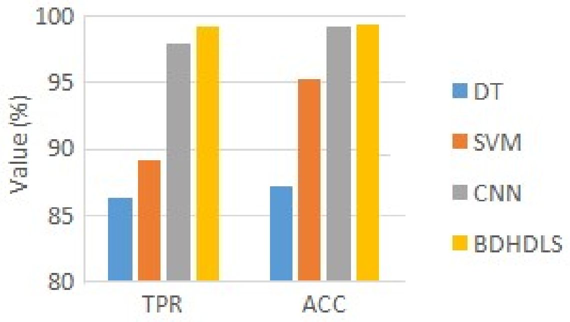

Fig. 15 Topaz Performance Ratio and Accuracy Check for the DARPA 1998 Dataset.

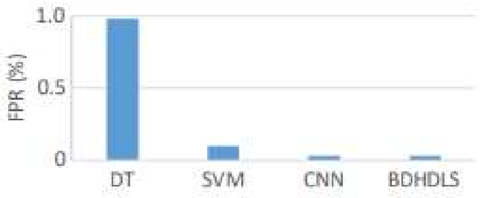

Fig. 16 Forward Probability of Success for the DARPA 1998 Dataset.

There aren't enough packets in each sample for RNN-CNN to work with the DARPA1998 dataset. In order to learn the temporal information of a lengthy series of data, LSTM does not fare well with a limited number of packets per sample. BDHDLS outperforms three other computational models in terms of classification accuracy.

5.4 Building times for BDHDLS with varying machine loads

Using the ISCX2012 dataset, Figure 17 displays the typical building time (in hours) for BDHDLS's 5 2 CV F test. When using all 64 computers, BDHDLS typically takes 12 hours to build. When more than one computer is used to build BDHDLS, the typical building time drops dramatically.

6 .Conclusion and Future Work

As part of this study, BDHDLS is presented with the goal of understanding the data distribution characteristics of certain invasive attack families. This method excels at capturing subtle data patterns for low-sample-size intrusion attempts. Similar to other DLS systems, BDHDLS takes on both behavioral and content-based innovations. In order to examine intrusive attack samples, BDHDLS takes into account both behavioral traits and content features of network traffic and the payload. Since IDS performance may be improved by combining the two kinds of characteristics, this strategy is an improvement over earlier methods. Moreover, we found that using big data methods and parallel strategies for feature selection, clustering, and training may drastically cut down on the time required to build a model. As a result, researchers may conduct more rapid iterations in their quest to find the optimal model parameters with which to solve their computing issues.

In this research, we use a simple decision fusion technique to combine the outputs of a number of concurrently running deep learning models. Although it's conceivable that this isn't the optimal method, it's possible that in the future we may try out more advanced decision fusion techniques, such as combining the outputs of many deep learning models in a tree structure. Rather of relying on humans to design merging methods, we want to test out deep neural networks to see whether we can integrate selections from many models in the tree. As feature selection and adoption has such a profound impact on the performance of deep learning models, we are also interested in incorporating new sets of behavioral traits into these frameworks to boost performance. Exploring the fast techniques for creating a multi-level cluster tree is crucial for cutting down on model development time. BDHDLS needs much more processing power to achieve the same performance gains as the baseline deep learning technique. Efforts will be made in the near future to reduce the amount of computational power required for the same level of performance.

References:

[1] Homeland Security Council, National strategy for homeland security, https://www.dhs.gov/xlibrary/assets/nat strat homelandsecurity 2007.pdf, 2007.

[2] S. Dua and X Du, Data Mining and Machine Learning in Cybersecurity. Boston, MA, USA: Auerbach Publications,2011.

[3] K. Kim and M. E. Aminanto, Deep learning in intrusion detection perspective: Overview and further challenges, in Proc. 2017 Int. Workshop on Big Data and Information Security (IWBIS), Jakarta, Indonesia, 2017, pp. 5–10.

[4] A. L. Buczak and E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor., vol. 18, no. 2, pp. 1153–1176, 2016.

[5] C. A. Catania and C. G. Garino, Automatic network intrusion detection: Current techniques and open issues, Comput. Electr. Eng., vol. 38, no. 5, pp. 1062–1072, 2012.

[6] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. Van Der Laak, B. Van Ginneken, and C. I. Sanchez, A survey on deep learning ´ in medical image analysis, Med. Image Anal., vol. 42, pp. 60–88, 2017.
[7] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, Shallow and deep networks intrusion detection system: A taxonomy and survey, arXiv preprint arXiv: 1701.02145, 2017.
[8] B. Chandra and R. K. Sharma, Deep learning with adaptive learning rate using laplacian score, Exp. Syst. Appl., vol. 63, pp. 1–7, 2016.
[9] Y. C. Li, X. Q. Nie, and R. Huang, Web spam classification method based on deep belief networks, Exp. Syst. Appl., vol. 96, pp. 261–270, 2018.
[10] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, vol. 521, no. 7553, pp. 436–444, 2015.
11] M. Papakostas and T. Giannakopoulos, Speech-music discrimination using deep visual feature extractors, Exp. Syst. Appl., vol. 114, pp. 334–344, 2018.
[12] Y. Yu, J. Long, and Z. P. Cai, Network intrusion detection through stacking dilated convolutional autoencoders, Secur. Commun. Networks, vol. 2017, p. 4184196, 2017.
[13] T. T. H. Le, J. Kim, and H. Kim, An effective intrusion detection classifier using long short-term memory with gradient descent optimization, in Proc. 2017 Int. Conf. Platform Technology and Service (PlatCon), Busan, South Korea, 2017, pp. 1–6.
[14] A. F. M. Agarap, A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data, in Proc. 10th Int. Conf. Machine Learning and Computing, Macau, China, 2018, pp. 26–30.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in Proc. 25th Int. Conf. Neural Information Processing Systems, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.
[16] 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.
[17] W. Wang, Y. Q. Sheng, J. L. Wang, X. W. Zeng, X. Z. Ye, Y. Z. Huang, and M. Zhu, HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection, IEEE Access, vol. 6, pp. 1792– 1806, 2017.
[18] E. Alpaydm, Combined 5 2 cv F test for comparing supervised classification learning algorithms, Neural Comput., vol. 11, no. 8, pp. 1885–1892, 1999.
[19] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, Assessing the accuracy of prediction algorithms for classification: An overview, Bioinformatics, vol. 16, no. 5, pp. 412–424, 2000.
[20] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerg. Top. Comput. Intell., vol. 2, no. 1, pp. 41–50, 2018.
[21] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, Network anomaly detection with the restricted boltzmann machine, Neurocomputing, vol. 122, pp. 13–23, 2013.
[22] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, vol. 61, pp. 85–117, 2015.

[23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access, vol. 7, pp. 41525–41550, 2019.

[24] S. M. Kasongo and Y. X. Sun, A deep learning method with filter based feature engineering for wireless intrusion detection system, IEEE Access, vol. 7, pp. 38597–38607, 2019.

[25] P. Nagar, H. K. Menaria, and M. Tiwari, Novel approach of intrusion detection classification deeplearning using SVM, presented at First International Conference on Sustainable Technologies for Computational Intelligence, Singapore, 2020, pp. 365–381.

[26] M. Akter, G. D. Dip, M. S. Mira, M. A. Hamid, and M. Mridha, Construing attacks of internet of things (IoT) and a prehensile intrusion detection system for anomaly detection using deep learning approach, presented at International Conference on Innovative Computing and Communications: Proceedings of ICICC 2019, Singapore, 2020, pp. 427–438.

[27] Z. Q. Liu, M. U. D. Ghulam, Y. Zhu, X. L. Yan, L. F. Wang, Z. J. Jiang, and J. C. Luo, Deep learning approach for ids, presented at Fourth International Congress on Information and Communication Technology: ICICT 2019, Singapore, 2020, pp. 471–479.

[28] C. Sekhar and K. V. Rao, A study: Machine learning and deep learning approaches for intrusion detection system, presented at Int. Conf. Computer Networks and Inventive Communication Technologies, Coimbatore, India, 2019, pp. 845–849.

[29] G. Nguyen, S. Dlugolinsky, V. Tran, and A. L. Garc´ıa, Deep learning for proactive network monitoring and security protection, IEEE Access, vol. 8, pp. 19696–19716, 2020.

[30] A. Abusitta, M. Bellaiche, M. Dagenais, and T. Halabi, A deep learning approach for proactive multi-cloud cooperative intrusion detection system, Future Generation Comput. Syst., vol. 98, pp. 308–318, 2019.

[31] A. Liu and B. Sun, An intrusion detection system based on a quantitative model of interaction mode between ports, IEEE Access, vol. 7, pp. 161725–161740, 2019.

[32] T. Aldwairi, D. Perera, and M. A. Novotny, An evaluation of the performance of restricted boltzmann machines as a model for anomaly network intrusion detection, Comput. Networks, vol. 144, pp. 111–119, 2018.

[33] C. Alliance, Big data analytics for security intelligence, https://downloads.cloudsecurityalliance.org/initiatives/bdwg/ Big Data Analytics for Security Intelligence.pdf, 2013.

[34] W. Zhong and F. Gu, A multi-level deep learning system for malware detection, Exp. Syst. Appl., vol. 133, pp. 151–162, 2019.

[35] J. W. Han and M. Kamber, Data Mining: Concepts and Techniques. San Francisco, CA, USA: Elsevier, 2011.

[36] S. K. Gupta, K. S. Rao, and V. Bhatnagar, K-means clustering algorithm for categorical attributes, in Proc. 1 st Int. Conf. Data Warehousing and Knowledge Discovery, Berlin, Germany: Springer, 1999, pp. 203–208.

[37] S. Owen, R. Anil, T. Dunning, and E. Friedman, Mahout in Action. Shelter Island, NY, USA: Manning Publications, 2011.

[38] W. Zhong, G. Altun, R. Harrison, P. C. Tai, and Y. Pan, Improved K-means clustering algorithm for exploring local protein sequence motifs representing common structural property, IEEE Trans. Nanobioscience, vol. 4, no. 3, pp. 255–265, 2005.

[39] L. D. Gibert, Convolutional neural networks for malware classification, Master dissertation, Universitat Politecnica ` de Catalunya, Tarragona, Spain, 2016.

[40] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in Proc. 2009 IEEE Symp. Computational Intelligence for Security and Defense Applications, Ottawa, Canada, 2009, pp. 1–6.

[41] J. Song, H. Takakura, and Y. Okabe, Description of Kyoto University benchmark data, http://www.takakura.com/ Kyoto data/BenchmarkData-Description-v5.pdf, 2006.

[42] R. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. E. Webster, and M. A. Zissman, Results of the DARPA 1998 offline intrusion detection evaluation, presented at Recent Advances in Intrusion Detection: 4th International Symposium, New York, NY, USA, 1999, pp. 829–835.

[43] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in Proc. 4 th Int. Conf. Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 2018, pp. 108–116.

[44] X. Chen, A simple utility to classify packets into flows, https://github.com/caesar0301/pkt2flow, 2017.

[45] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, Network anomaly detection: Methods, systems and tools, IEEE Commun. Surv. Tutor., vol. 16, no. 1, pp. 303–336, 2014.