BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 03/06 pp181–195
Volume 3, Number 3, September 2020
DOI: 10.26599/BDMA.2020.9020003

Applying Big Data Based Deep Learning System to Intrusion Detection

Wei Zhong*, Ning Yu, and Chunyu Ai

Abstract: With vast amounts of data being generated daily and the ever increasing interconnectivity of the world's internet infrastructures, a machine learning based Intrusion Detection Systems (IDS) has become a vital component to protect our economic and national security. Previous shallow learning and deep learning strategies adopt the single learning model approach for intrusion detection. The single learning model approach may experience problems to understand increasingly complicated data distribution of intrusion patterns. Particularly, the single deep learning model may not be effective to capture unique patterns from intrusive attacks having a small number of samples. In order to further enhance the performance of machine learning based IDS, we propose the Big Data based Hierarchical Deep Learning System (BDHDLS). BDHDLS utilizes behavioral features and content features to understand both network traffic characteristics and information stored in the payload. Each deep learning model in the BDHDLS concentrates its efforts to learn the unique data distribution in one cluster. This strategy can increase the detection rate of intrusive attacks as compared to the previous single learning model approaches. Based on parallel training strategy and big data techniques, the model construction time of BDHDLS is reduced substantially when multiple machines are deployed.

Key words: intrusion detection; deep learning; convolution neural network; fully connected feedforward neural network; multi-level clustering algorithm

1 Introduction

With vast amounts of data being created every day and the ever increasing interconnectivity of the world's internet infrastructures, more and more security vulnerabilities in these infrastructures are discovered by security experts every month[1]. These security vulnerabilities create opportunities for cyber criminals to intrude into these infrastructures and perform malicious activities[2]. Consequently, cyber security experts and designers need to develop various intrusion detection systems to protect computers and networks from hackers who may attack the network system and steal and/or destroy financial, medical, or other valuable information from databases[2].

The traditional Intrusion Detection System (IDS) usually utilizes precise description such as rules or signatures to monitor traffic[3]. The signature based approach generally has a low false positive rate. Since new intrusion techniques are being designed and invented every day, experts need to frequently update the database containing these rules and signatures, which is a labor-intensive process. Sometimes it is very challenging to develop appropriate signatures for more sophisticated attacks which are evolved from previous attacks[3].

Besides traditional intrusion detection systems, machine learning techniques have been explored by researchers[4]. Compared with traditional intrusion detection systems, machine learning techniques can

• Wei Zhong and Chunyu Ai are with the Division of Math and Computer Science, University of South Carolina Upstate, Spartanburg, SC 29303, USA. E-mail: wzhong@uscupstate.edu; aic@uscupstate.edu.
• Ning Yu is with the Department of Computing Sciences, State University of New York College at Brockport, Brockport, NY 14420, USA. E-mail: nyu@brockport.edu.
* To whom correspondence should be addressed.
Manuscript received: 2020-03-08; revised: 2020-03-27; accepted: 2020-03-30

automatically reason about intrusive and benign samples to fit the most appropriate detection model parameters. Particularly, machine learning techniques can be used to discover patterns of complex intrusive activities after examining features represented by the network traffic[5]. In recent years, the large number of labeled samples, powerful computational hardware, and breakthrough in machine learning algorithms have triggered research community to re-evaluate the significance of machine learning techniques for intrusion detection[3–6]. The shallow learning model and the deep learning model are two main types of machine learning techniques for intrusion detection systems[3, 4]. The shallow learning model typically consists of less than three computational layers[3]. Examples of the shallow learning model include support vector machine, logistics regression, and decision tree[7]. A single shallow learning model is normally constructed on an entire dataset. A single shallow learning model may encounter difficulty to discover useful patterns from a large number of training samples. Researchers have shown that there are mapping functions that the shallow learning model cannot learn[7]. In general, the shallow learning model has not accomplished satisfactory performance for intrusion detection since the shallow learning model is not effective to take advantages of patterns offered by the increasing number of samples.

In the past ten years, deep learning is one of the most important technical breakthroughs in the artificial intelligence field. In contrast to the shallow learning, deep learning generally requires a large number of neural layers[8, 9]. Deep learning has produced better results than shallow learning models in the fields of computer vision, speech recognition, automatic machine translation, and finance[10, 11]. Advantages of deep learning in other areas have inspired researchers to apply deep learning to intrusion detection[12–15]. Conventionally, a single deep learning model is constructed on the whole dataset. The single deep learning model generally works very well when huge amount of samples are available, such as computer vision[10]. Under the situation where huge amount of samples are not available, the single deep learning model may have troubles[10]. For the intrusion detection task, the single deep learning based approach may experience problems to understand increasingly complicated data distribution of intrusion samples. Particularly, the single deep learning model may encounter difficulty to capture unique patterns from the intrusive attacks having a small number of samples.

To further enhance performance of a single shallow learning model and a single deep learning model, we propose a Big Data based Hierarchical Deep Learning System (BDHDLS) that organizes multiple deep learning models using the hierarchical tree structure. The hierarchical tree structure is developed to partition samples into multiple level clusters and each deep learning model trained on related samples in one cluster is adapted to study the unique data distribution for that cluster. The construction of BDHDLS has five phases. In the first phase, behavioral features and content features are extracted and selected using big data techniques. In the second phase, the entire dataset is partitioned into one-level clusters by the parallel improved K-means clustering approach. Samples in each cluster of the tree generally display similar traffic patterns[4, 5]. In the third phase, the hierarchical clustering process is carried out for each one level cluster with low quality in parallel to produce the cluster subtree. Then these subtrees are combined to generate multiple-level hierarchical trees of clusters. In other words, the entire dataset is partitioned into multiple clusters in multiple levels. In the fourth phase, the deep learning model is built for each cluster in the hierarchical tree to learn the distinctive data distribution pattern for that cluster. In the final phase, decision values of deep learning models are merged to make the final judgement about whether the test sample is intrusive or not.

Feature extraction and selection have big impact on performance of deep learning models. In our proposed model, both behavioral features and content features are adopted. Behavioral features and content features are used to analyze traffic patterns in different angles. Behavioral features describe network traffic characteristics including source-destination ports/IP addresses, various packets-level/flow-level statistics, and durations of connections. The content features describe the subtle patterns embedded into the payload information[16]. Combining both types of features can improve robustness of the learning algorithms. In order to evaluate different approaches for building the intrusion detection system, the performance of five computational models is compared: (1) a single Decision Tree (DT) built on the entire dataset; (2) a single Support Vector Machine (SVM) built on the entire dataset; (3) a single deep Convolutional Neural Network (CNN) built on the entire dataset; (4) a single model (RNN-CNN) combining Recurrent Neural Network (RNN) and

CNN built on the entire dataset; and (5) BDHDLS built on the tree structure. SVM and DT are the shallow learning architecture. CNN is based on the HAST-I architecture described in Ref. [17]. CNN-RNN is based on the HAST-II architecture described in Ref. [17]. CNN and CNN-RNN extract content features from payloads automatically. Both the combined 5×2 Cross Validation (CV) F test[18] and the independent test are performed to evaluate the performance of the intrusion detection system. The evaluation metrics used in this work include True Positive Rate (TPR) or detection rate, False Positive Rate (FPR) or false alarm rate, and accuracy[3, 19].

Major contributions of this work include (1) utilization of big data techniques called Apache Spark for feature selection and clustering; (2) incorporation of both behavioral and content-based features simultaneously to improve prediction accuracy; and (3) adoption of multiple deep learning models in the hierarchical tree structure to learn unique traffic patterns for each intrusive attack family.

The rest of the paper is organized as follows. Section 2 discusses the related works. Section 3 describes five phases of BDHDLS. Section 4 explains the training set, the testing set, evaluation metrics, and evaluation methods. Section 5 presents the experimental results and analysis. Finally, Section 6 concludes the paper and points out the future work.

2 Related Work

Traditional signature-based approaches dominate intrusion detection softwares since they achieve a low false positive rate[2]. The signature-based approaches can detect attacks by searching specific patterns including a set of behavioral sequences in network traffic or well-known malicious instructions in the payload of packets[2]. Since the complexity of network attacks continues to grow rapidly, constant updates of the signature database are required. These updates are generally labor intensive. As a result, the lag between discovery of a new attack strategy and signature updates can potentially result in failure of identifying new network threats[1, 2]. Furthermore, the signature-based approach may not discover attacks with slightly modified payloads[1, 2].

In order to avoid potential weaknesses of traditional signature-based approaches, researchers have employed the shallow learning models having less than three computational layers such as DT and SVM for intrusion detection[2]. These shallow learning models generally cannot further improve their performance with the increasing number of training samples since the shallow learning models struggle to explore a rich set of patterns provided by a large number of training samples. Since a deep learning model can keep boosting their performance with more and more training samples as compared to a shallow learning model[3], researchers have experimented on a single deep learning model using the whole dataset for intrusion detection[12–14, 20, 21].

Constructing a single learning model for a dataset that combines samples from multiple types of intrusive attacks has three disadvantages: (1) problem to learn complicated data distribution; (2) difficulty to capture patterns from intrusive attacks having a small number of training samples; and (3) scalability. Firstly, each type of intrusive attacks has distinctive characteristics, strategy, and data distribution[2]. Additionally, the variety and sophistication of intrusive attacks keep growing quickly due to new attacking strategies that are created each year[2]. As a result, combining different types of intrusive attacks into one dataset can increase the complexity of the entire dataset significantly and make the learning task more challenging[2]. Secondly, some types of intrusive attacks only have a small number of samples. After merging these types of intrusive attacks having a small number of training samples into the entire large dataset, it is difficult for the deep learning model to learn patterns of these intrusive attacks. Thirdly, constructing a single deep learning model is not scalable to study patterns of increasingly big intrusion detection datasets. Training deep learning models with millions of samples may require several weeks to complete[22]. The expensive training cost may hamper efforts to explore different deep learning architectures quickly so that researchers can adjust detection strategies and techniques in order to respond to new variants of attacks.

Choosing the appropriate set of features is another critical issue for machine learning algorithms. Behavioral features and content features are two main types of features adopted by machine learning based IDS[2]. These two types of features are used to analyze the pattern of the intrusive activities in different angles. The behavioral features focus on network traffic characteristics, such as source-destination ports/IP addresses, various packets-level/flow-level statistics, and duration of collections. Various deep neural networks were developed in recent literatures to evaluate the network behaviors[23–32]. In the contrast, content features are used to uncover patterns of

intrusions through the content of payloads in the traffic. Deep examination of payloads may reveal valuable information related to exploit methods, malicious actions, or illegal commands[16]. Both behavioral feature based approaches and content feature based approaches have their advantages and disadvantages. Behavioral feature based approaches may not discover all intrusive activities since behavioral features only capture network traffic characteristics. Content feature based approaches are the powerful tools to recognize intrusive activities. However, content feature based approaches may suffer from payload obfuscation, which may be achieved by polymorphism and metamorphism. In the meantime, it is difficult for the attacker to obfuscate its behavioral patterns[2]. Most of current researches use either behavioral features or content features. To the best of our knowledge, no research has combined behavioral and content features systematically and intelligently. For content feature extraction, the scalability is another major issue. Since billions of candidate content features can be extracted from payloads, big data techniques are suitable to handle this overwhelming amount of data[30, 33].

To overcome potential weaknesses of previous approaches, we propose a BDHDLS for intrusion detection. In the proposed BDHDLS, each deep learning model in the tree structure is constructed on the data subset for a particular group of intrusive attack families. Hence, each model is specialized to study the data distribution for a specific group of intrusive attacks and all deep learning models in the hierarchical tree cooperate to make the final decision. As compared to previous approaches for intrusion detection, BDHDLS has better chances to learn unique data distribution patterns for each intrusive attack family. As compared to previous approaches who only use either behavioral features or content features, BDHDLS considers both behavioral features and content features to enhance the capability to understand increasingly complex intrusion data distribution. In order to address IDS scalability issues, BDHDLS adopts big data techniques to extract features and parallel strategies to accomplish clustering and model training.

This work is motivated by the multi-level deep learning model for malware detection[34]. As compared to malware detection, this work for intrusion detection creates unique challenges and requires different strategies to select useful features and build multi-level clusters. Particularly, feature selection and clustering process require utilization of big data techniques in order to speed up the time and space intensive computational process. In this work, selection of appropriate deep learning architecture and its hyper parameter is highly dependent on feature extraction from behavioral and content characteristics from the network traffic. As a result, the process of deep learning training and parameter tuning in this work is quite different from the model construction process for malware detection.

3 BDHDLS for Intrusion Detection

Construction of BDHDLS is divided into five phases: Phase 1: Generating behavioral features and content features using big data techniques; Phase 2: Partitioning the dataset into multiple one-level clusters using Spark based parallel improved K-means algorithm; Phase 3: Generating multi-level cluster trees in parallel; Phase 4: Building the deep learning model for each cluster; Phase 5: Merging decisions from deep learning models in different clusters to classify samples as intrusive or benign. The flow-chart to construct BDHDLS is shown in Fig. 1.

3.1 Generation of behavioral features and content features

Since different choices of features can impact the performance of machine learning algorithms significantly, feature generation and extraction are one of the most important tasks for any machine learning model. In this section, the process of producing behavioral features and content features is discussed.

At first, behavioral features are generated after analyzing network traffic. Behavioral features in this work include the number of data bytes from source to destination, various packets-level/flow-level statistics, type of the protocol, and duration of collection[16]. In other words, behavioral features are used to reflect characteristics of network traffic patterns.

After generating behavioral features, content features are extracted from payloads in order to uncover vital information embedded into the payload, which is the primary location for intruders to initiate a set of malicious instructions, exploit methods, or illegal commands[16]. In the first step of content feature extraction, n-grams (n-byte sequences) are generated by sliding an n-byte window. After payloads for all samples in the whole dataset are processed, frequencies of unique n-grams appearing in the benign samples and intrusive samples are calculated, respectively. It

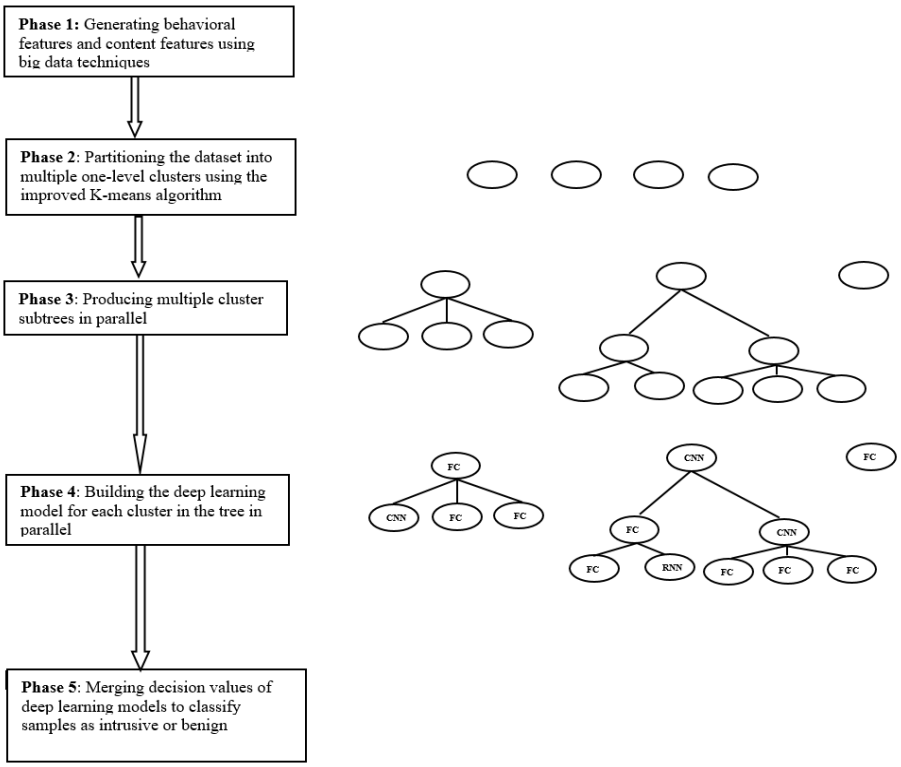

Fig. 1 Flow chart for building BDHDLS, where FC represents Full Connected feed forward neural network.

is generally inappropriate to use all byte n-gram features extracted from payloads. When the classifier is trained with the exponential number of such binary n-gram features, the effectiveness of a classifier can be reduced significantly since most of these features may be noisy, redundant, or irrelevant. In order to avoid these problems, candidate binary n-gram features need to be sorted based on certain criterion so that a small subset of features with the greatest discriminatory power can be selected for model construction.

The information gain is used as the selection criterion for important content features in this work since it is one of the most effective feature selection measures reported in literatures[2, 35]. In this work, the information gain is utilized to measure the effectiveness of a feature to classify the training data. If the training data is split based on values of this feature, information gain measures the expected reduction in entropy after the split. The feature with higher information gain is more effective to classify the samples. The information gain can be defined as Eq. (1)[35]:

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

where $\text{Info}(D)$ is the original information requirement and $\text{Info}_A(D)$ is the new requirement after splitting on the feature A. $\text{Info}(D)$ is defined as Eq. (2):

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

where pos is the total number of positive (intrusive) samples in I, total is the total number of samples in I, and neg is the total number of negative (benign) samples in I. I represents the whole dataset in Eq. (2). $\text{Info}_A(D)$ is defined as Eq. (3):

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

The feature extraction and selection process can be time consuming and space intensive for very large datasets. In order to provide the efficient and effective solution for feature selection, the Spark (in-memory MapReduce) framework is utilized to perform parallel tasks for byte n-gram extraction and selection in the cluster of machines. The Spark (in-memory MapReduce) framework can process large datasets in parallel, distributing the workload across large clusters of commodity machines.

Figure 2 shows major steps to extract important content features using the Spark framework. Based on the Spark framework, the training dataset samples are evenly distributed among m nodes in the cluster. Byte

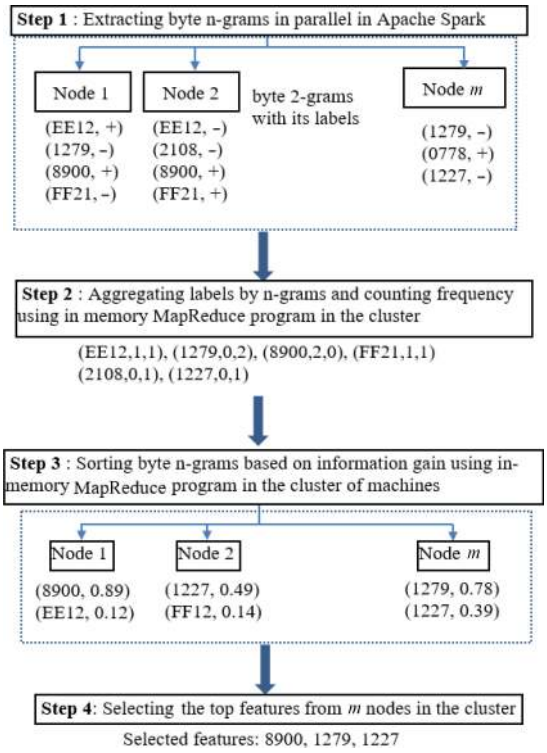

Fig. 2 Major steps to extract important content-based features using the Spark framework.

n-grams are extracted from each training file in parallel, using m nodes in the cloud cluster (Fig. 2, Step 1). For example, byte 2-gram EE12 is observed once in one positive (intrusive) sample in Node 1, and meanwhile it is observed once in a negative (benign) sample in Node 2. This is denoted by (EE12, $+$) and (EE12, $-$) under Nodes 1 and 2, respectively, in Step 1 of Fig. 2. In Step 2 of Fig. 2, the in-memory MapReduce program first aggregates labels of identical n-grams. As a result, the aggregated pair for EE12 is (EE12, $+, -$). The aggregated n-grams are distributed to reducers in the m nodes of the cloud cluster. Each reducer counts aggregated labels to obtain a positive count and a negative count. The positive count shows how many intrusive samples contain the particular feature. The negative count shows how many benign samples contain the particular feature. This is denoted by (EE12, 1, 1) in Step 2 of Fig. 2. In Step 3 of Fig. 2, the in-memory MapReduce program calculates the information gain of each byte n-gram feature and sorts these byte n-gram features based on the information gain. In Step 4 of Fig. 2, the reduce phase selects the best feature based on the information gain using one node while the map phase does nothing. Each feature in a content feature set is a binary feature. If the given content feature is present in a sample, its value is 1; otherwise, its value is 0.

3.2 Generation of one-level clusters

After behavioral and content features are generated, these features are used to produce one-level clusters with the K-means algorithm. Initialization techniques and scalability are two major issues for the traditional K-means algorithms. Traditional K-means algorithms[36] randomly select samples as initial cluster centers. This strategy can result in distorted or improper partitions, which may be deviated from globally optimal solutions. For instance, a small number of clusters may capture a large percentage of samples and remaining clusters may gather a few number of samples due to random choice of initial cluster centers. Traditional K-means algorithms also face the challenge to process large number of samples efficiently[36].

In order to tackle issues of random selection and scalability, the parallel improved K-means clustering using Apache Spark[37] is developed in this work. The parallel K-means algorithm adopts the greedy initialization techniques[38] to overcome weaknesses of random initialization. The goal of the greedy initialization technique is to choose proper initial points, which can lead to the final clustering solution representing the underlying data distribution more accurately and consistently.

Each iteration of the parallel K-means clustering has two phases: (1) the mapping phase and (2) reducing phase. In the mapping phase, the closest centroid for each sample located in different data partitions is computed in parallel. Each data partition of the dataset is placed in different machines. In the reducing phase, the centroid for each cluster is recomputed after acquiring the partial sum for the centroid from each data partition for one cluster. This divide-and-conquer strategy can allow the clustering algorithm to handle millions of samples efficiently. The distance score between the sample and the cluster center is defined as Eq. (4):

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

where N is the number of features for each sample, $F_x(i)$ is the value of the feature at index i for the sample x, and $F_c(i)$ is the value of the feature at index i for the centroid of the given cluster.

3.3 Generation of hierarchical based cluster tree

The goal of one-level clusters carried out by the improved K-means clustering is to group samples with

similar patterns into the same cluster. Analysis of one-level clustering demonstrates that some of these one-level clusters have low quality. The quality of the cluster is defined as Eq. (5):

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

where P_{benign} is the percentage of benign samples in the given cluster and $P_{\text{intrusive}}$ is the percentage of samples belonging to intrusive samples in the given cluster. For example, if the percentage of benign samples in the given cluster is 20% and the percentage of intrusive samples in the same cluster is 80%, the quality of this cluster is 80%.

The parallel multi-level clustering is implemented to find out the high quality subclusters from one-level clusters having low quality. The parallel multi-level clustering has 2 steps: (1) local discovery step and (2) merging step. In the local discovery step, the agglomerative hierarchical clustering[35] is performed on each one-level clusters having low quality. During this process, subclusters in these one-level clusters continue to merge until the quality of the merged clusters drops below the given threshold. At the end of the local discovery step, a forest of cluster subtrees is created. In the merging step, the root clusters of each subtree are combined until the quality of the merged clusters drops below the given threshold. In this work, the multi-level tree structure is used to learn unique underlying data distribution patterns of a particular sample subspace in the given level of the tree.

3.4 Deep learning model training for each cluster

The hierarchical clustering algorithm can discover some high-quality subclusters from one-level clusters. Since it is difficult to accurately define the distance function for calculating similarity between samples, the noisy and irrelevant information can still be incorporated into each cluster in the multi-level hierarchical tree structure. These noisy and irrelevant information can considerably decrease the effectiveness of the intrusion detection system.

In order to improve the intrusion detection performance of the hierarchical tree structure, the deep learning model is trained for each cluster in the multi-level cluster tree based on combination of behavioral features and content features. Three different types of deep learning models including FC, CNN, and RNN are evaluated in order to select the best model for each cluster in the multi-level cluster tree.

The deep FC has many layers of computational units interconnected in a feed-forward fashion. FC is the most general deep learning model to figure out transformation functions. Since architecture of FC does not take the structure of data into consideration, it may face difficulty to handle data with special characteristics.

CNN focuses on determining spatial patterns of traffic byte information from the payload, which can be represented as the matrix of zeros and ones[17]. CNN contains many convolutional and pooling layers, several dense layers, and the softmax output layer[39]. Figure 3 illustrates the architecture of the deep CNN. Compared to the regular deep fully connected network, the CNN has two major characteristics: (1) sparsity of connection and (2) weight sharing. These two characteristics allow CNN to extract features from the raw data automatically. As the computational layers go deeper and deeper, the CNN can gradually combine simple features in the earlier layers into more complex features in the later layers after uncovering the relationship between neighboring input feature vectors[39]. The CNN is particularly robust against code obfuscation and instruction reordering in the payload since CNN can find out lower-to-higher order local features that are intrinsic for the functionality of intrusive attacks.

RNN is the special type of neural network. Taking sequential information into consideration, each output of RNN depends on previous outputs. RNN is specifically built to examine the sequence of payloads in one sample in order to learn the sequential dependence[17]. Figure 4 illustrates the architecture of the deep RNN.

With combination of behavioral features and content features, the deep learning model is designed to analyze both patterns of network traffic characteristics and distinctive information in the payload of the traffic sample for each cluster. Each deep learning model can concentrate on highly similar samples in each cluster

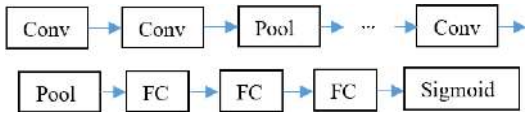

Fig. 3 Diagram for CNN.

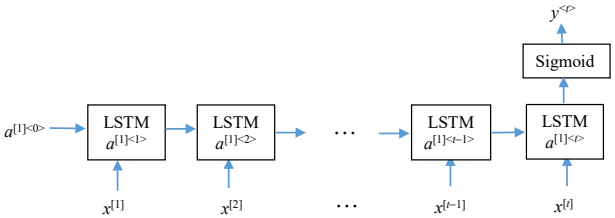

Fig. 4 Diagram for RNN.

without being distracted by unrelated information from other clusters. As a result, the learning efficiency for each deep learning model can be enhanced.

The core of deep FC, CNN, and RNN is the function mapping from the input vector to the output vector for a large number of layers. The mapping function for each layer of the deep learning model is defined as Eq. (6)[39]:

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

where a_i is the activation of the i-th layer, W_i is the weight matrix for i-th layer, b_i is the bias for i-th layer, and f_i is the activation function. The loss function is used to measure the difference between the real output and the predicted output. The loss function for all 3 types of neural networks is defined as Eq. (7)[39]:

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

where y_i is the true label for sample i, $y_i \in 0, 1$ with 0 representing benign sample and 1 malware. $\hat{y_i}$ is the output of our deep learning model for sample i. M is the size of one batch.

Based on the loss function, the cost function provides important guidance about how to optimize parameters of deep learning models. The cost function is defined as Eq. (8):

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

where $J(w_1, b_1, \ldots)$ represents the optimization cost in respect of all weights and bias.

The formula to adjust the network parameters during one iteration of gradient descent is defined as Eq. (9):

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

where α is the learning rate.

3.5 Decision fusion algorithm

The deep learning model's decision function for the cluster k to classify sample x as benign or intrusive is expressed as Eq. (10):

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

where y_k is the "activation" of the last layer of the neural network for the cluster k and W_k is the weight matrix between the last layer and the output layer of neural networks for cluster k. The decision value reveals how confident the deep learning model predicts the sample as benign or intrusive.

Deep learning models are constructed in different clusters with quite different data distributions. In order to compare classification values from different models fairly and objectively, the classification value, $f_k(x)$, of the deep learning model is normalized using the z-score[35]. This normalization process is necessary, because decision functions of deep learning models for different clusters are calculated from various high-dimensional feature spaces formed by related samples belonging to the same cluster[39]. The normalized decision value of the cluster k for sample x is defined as Eq. (11):

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

where mean_k is the mean classification values of the deep learning model in cluster k and δ_k is the standard deviation of classification values of the deep learning model for the cluster k. Higher decision value shows that the deep learning model is more confident to classify a given sample.

Since the distance between the sample and the cluster associated with this deep learning model can affect the confidence level of the decision value, the deep learning decision value for the cluster is weighted by the distance between the sample and the cluster associated with the deep learning model. The distance function is defined as Eq. (12):

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

where $F_x(i)$ is the value of features at index i for the sample x and $F_k(i)$ is the value of features at index i for the centroid of cluster k. The logistic function used to smooth the distance between the sample x and cluster k is defined as Eq. (13):

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

where k is the cluster k and x is the given sample. As a result, the weighted decision value for a sample x is defined as Eq. (14):

$$\psi(k, x) = \text{decision\_value}_k(x) \times \text{smooth\_dist}(k, x) \tag{14}$$

If the training accuracy of the deep learning models for a given cluster falls below the given threshold, these models will be excluded from the decision making process since these models have low quality. Finally, the highest weighted decision value is utilized to classify samples. Figure 5 shows the detailed algorithm to build BDHDLS.

4 Datasets and Experimental Setup

In this section, datasets for the combined 5×2 CV F test and independent test are discussed first, then the details

BDHDLS

Phase 1. Generating behavioral features and content features using big data techniques.

Phase 2. Separating the entire dataset into several one-level clusters based on the parallel improved K-means clustering algorithm. The value of K is carefully chosen based on repeated experiments for each dataset.

WHILE (the number of initial points found is below the required number of clusters)
{
Randomly choose K initial points.
Run the parallel K-means algorithm in Apache Spark Infrastructure for several iterations.
Evaluate the quality of clusters produced by each initial point.
Mark initial points which produce the high quality clusters as the candidate points.
Include the candidate points whose minimum distance to existing points in the initialization array are above the threshold into the initialization array.
}
END WHILE

Execute the parallel K-means algorithm with the carefully chosen initial points in the Apache Spark Framework.

Phase 3. Producing multi-level cluster tree.

FOR each one-level cluster with the low quality
{
Performing the agglomerative hierarchical clustering in parallel. During this process, subclusters in these one level clusters continue to merge until the quality of the merged clusters drops below the given threshold. This step basically generates hierarchical based subtrees for this cluster.
}

Combining root clusters of each subtree until the quality of the merged clusters drops below the given threshold. This step produces the multi-level cluster tree.

Phase 4. Training the deep learning model for each cluster in the cluster tree in parallel.

Phase 5. Selecting the most confident deep learning model from the cluster tree and classifying samples as benign or intrusive.

Adjusting the decision value of the deep learning model by the distance score

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

Fig. 5 Five phases of BDHDLS.

of performance metrics and model configurations are explained.

4.1 Datasets for combined 5 × 2 cross validation F test and independent test

The dataset used to build the intrusion detection model needs containing a comprehensive and extensive set of intrusion with the complete profile of real-time background traffic[16]. The dataset also needs providing the raw data with the payload information. The raw data with the payload information is very important for providing a clear and complete information about how intrusive attacks may affect the network and how the servers can respond to such an intrusion[16]. The popular public datasets such as NSL-KDD[40] and Kyoto2009[41] do not contain raw traffic data. DARPA1998[42], ISCX2012[16], and CICIDS2017[43] are only public benchmark datasets containing raw traffic data with both labeled benign and intrusive samples. Each sample in this work is defined as one network flow, consisting of multiple network packets communicated between 2 sides. These traffic bytes from multiple packets are combined to form one sample.

The DARPA1998 dataset consists of benign samples and four types of intrusive samples including DoS, Probe, U2R, and R2L[42]. The DARPA1998 dataset contains 3.5 million samples. The ISCX2012 dataset is the public benchmark dataset which captures the complete network interaction and payload information. It contains various multi-stage attacks with realistic background traffic[16]. The ISCX2012 dataset has 1.4 million benign samples and around 41 000 intrusive samples. Four types of intrusive samples in the ISCX2012 dataset include BFSSH, DDos, HttpDos, and Infiltrating, which is more realistic than the DARPA1998 dataset[16]. The CICIDS2017 dataset is the latest dataset that contains benign and intrusive samples. The type of intrusive samples in the CICIDS2017 dataset include Botnet, Web Attack XSS, Web Attack BF, Patator, Port Scan, and Dos[43]. The CICIDS2017 dataset has 2.8 million samples.

In this work, several computational models are tested using these three datasets, respectively. For each dataset, 80% of samples are randomly selected for model training and the combined 5 × 2 cross validation F test. The remaining 20% of samples serve as the independent testing set.

During data preprocessing, the pkt2flow tool[44] is used to split raw pcap files into multiple network flows.

4.2 Performance evaluation metrics

True positive rate, false positive rate, and accuracy are adopted to evaluate the performance of the intrusion detection system. TPR is defined as Eq. (15)[45]:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{15}$$

where True Positive (TP) represents the number of intrusive samples that are correctly recognized and False Negative (FN) represents the number of intrusive samples that are incorrectly recognized as benign samples. TPR is also called the detection rate. FPR is defined as Eq. (16)[45]:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{16}$$

where False Positive (FP) represents the number of benign samples that are incorrectly recognized as intrusive samples and True Negative (TN) represents the number of benign samples that are correctly recognized. FPR is also called the false alarm rate. The accuracy is defined as Eq. (17):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{17}$$

4.3 Model configuration for different clusters in the subtree

In Table 1, model configurations of CNN are explained in details. In Table 2, model configurations for FC are represented. In Table 3, model configurations for RNN are shown.

The cross-validation technique is used to choose the best model configuration. Leaky Relu is selected as activation functions for all models. The dropout rate is set to 0.1 to avoid overfitting. The combined 5×2 CV F test is adopted to choose the most suitable model configuration for each cluster.

Table 3 Model configurations of recurrent neural network.

Hyper parameter	Depth	Number of neurons
LSTM-1	1	128
LSTM-2	2	256

5 Experimental Result and Analysis

In this section, experimental results for the combined 5×2 CV F test and the independent test are reported for three datasets.

5.1 Performance comparison of different feature sets

At first, performance of different feature sets are compared. Figure 6 compares TPR and accuracy (ACC) for different feature sets in the ISCX2012 dataset. Figure 7 compares FPR for different feature sets in

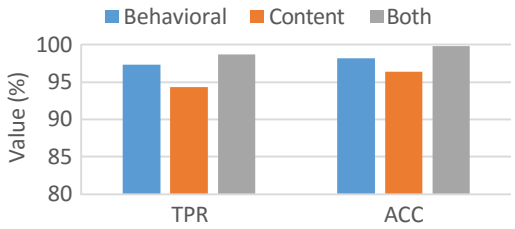

Fig. 6 TPR and ACC for different feature sets in the ISCX2012 dataset.

Table 1 Model configurations of convolutional neural network.

Model configuration 1	Model configuration 2	Model configuration 3
1 Conv layer (64 5×5 filters)	3 Conv layers (64 5×5 filters)	5 Conv layers (64 5×5 filters)
Max-pooling layer	Max-pooling layer	Max-pooling layer
1 Conv layer (128 5×5 filters)	3 Conv layers (128 5×5 filters)	5 Conv layers (128 5×5 filters)
Max-pooling layer	Max-pooling layer	Max-pooling layer
1 Conv layer (256 5×5 filters)	3 Conv layers (256 5×5 filters)	5 Conv layers (256 5×5 filters)
Max-pooling layer	Max-pooling layer	Max-pooling layer
1 FC layer (1024 neurons)	3 FC layers (1024 neurons)	5 FC layers (1024 neurons)
Sigmoid output layer	Sigmoid output layer	Sigmoid output layer

Table 2 Model configurations of FC.

Number of layers	Number of neurons
8	[128, 128, 128, 128, 64, 64, 64, 64]
8	[64, 64, 64, 64, 128, 128, 128, 128]
12	[256, 256, 256, 256, 128, 128, 128, 128, 64, 64, 64, 64]
12	[64, 64, 64, 64, 128, 128, 128, 128, 256, 256, 256, 256]
16	[256, 256, 256, 256, 128, 128, 128, 128, 64, 64, 64, 64, 32, 32, 32, 32]
16	[32, 32, 32, 32, 64, 64, 64, 64, 128, 128, 128, 128, 256, 256, 256, 256]

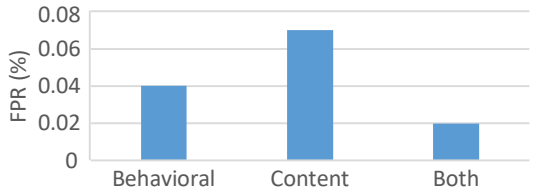

Fig. 7 FPR for different feature sets in the ISCX2012 dataset.

the ISCX2012 dataset. Our results demonstrate that combination of both behavioral features and content features performs best. Comparison of different feature sets in other two datasets shows similar trends.

5.2 Results for ISCX2012 dataset

In this section, the independent test results for the ISCX2012 dataset are discussed. Then the statistical analysis for the combined 5 × 2 CV F test for the ISCX2012 dataset is performed.

5.2.1 Independent test results for ISCX2012 dataset

Figure 8 shows the number of samples for different intrusive attacks in the ISCX2012 dataset.

In Fig. 9, TPR and ACC for the ISCX2012 dataset were presented. In Fig. 10, false positive rate (false alarm rate) for five models is compared. The implementation of CNN is based on the HAST-I model and implementation of CNN-RNN is based on the HAST-II model[17]. Compared with the CNN-RNN model, BDHDLS improves TPR by 2.5 percentage points. The FPR of

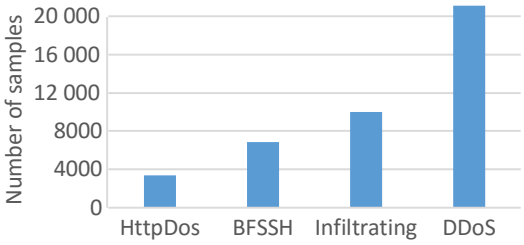

Fig. 8 Number of samples of different intrusive attacks for ISCX2012 dataset.

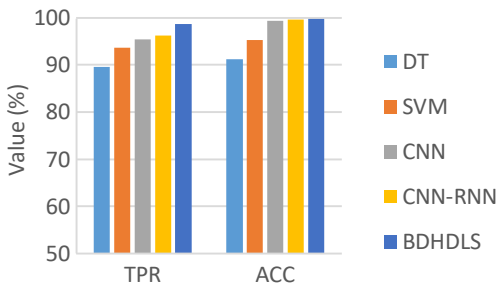

Fig. 9 TPR and ACC for ISCX2012 dataset.

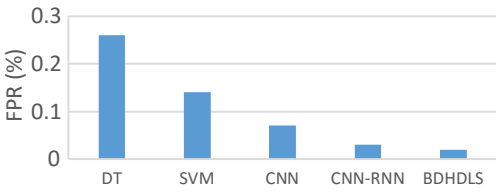

Fig. 10 FPR for ISCX2012 dataset.

BDHDLS is comparable to other computational models. Figures 9 and 10 also show that in general deep learning models perform better than the shallow learning models. As a result of hierarchical learning structure adoption, the performance of BDHDLS is better than other single deep learning approaches.

5.2.2 Statistical analysis of intrusion detection performance for the 5 × 2 CV F test in the ISCX2012 dataset

Besides the independent testing carried out in the testing set, the combined 5 × 2 CV F test is performed in the training set to determine whether the intrusion detection performance improvement of BDHDLS over the other four models in the ISCX2012 dataset is statistically significant.

The p-value produced by the combined 5 × 2 CV F test specifies the significant level at which the null hypothesis that algorithms have the same error rate can be rejected. Lower p-value usually implies a more statistically significant improvement of BDHDLS over the other four models. In this work, the significant level for p-value is set to 1%, which is more rigorous than 5% typically chosen by statistician. Table 4 shows "p value by F test" when binary classification of five computational models are performed. Experimental results from Table 4 indicate that performance gains of BDHDLS over the other four computational models in terms of all evaluation metric are statistically significant.

5.3 Results for CICIDS2017 dataset

In this section, the independent test results for the CICIDS2017 dataset are discussed. Then the statistical analysis for the combined 5 × 2 CV F test for the

Table 4 "p value by F test" for binary classification in the ISCX2012 dataset. (%)

Model	FPR	TPR	ACC
DT	<0.1	<0.1	<0.1
SVM	<0.1	<0.1	<0.1
CNN	<0.1	0.2	0.7
RNN-CNN	0.8	0.3	0.9
BDHDLS	N/A	N/A	N/A

CICIDS2017 dataset is conducted.

5.3.1 Independent test results for CICIDS2017 dataset

Figure 11 indicates that the number of samples for different intrusive attacks in the CICIDS2017 dataset.

The following results are obtained from the independent test dataset. In Fig. 12, TPR and ACC for the independent testing in the CICIDS2017 dataset are presented. In Fig. 13, the false positive rate (false alarm rate) in the CICIDS2017 for five models is compared. The BDHDLS improves the TPR by around 2 percentage point as compared to CNN-RNN.

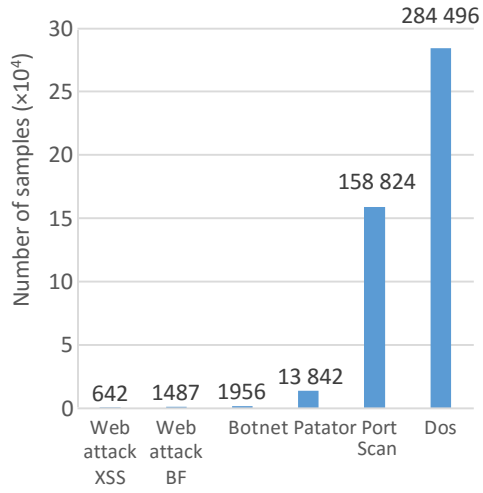

Fig. 11 Number of samples for different intrusive attacks in the CICIDS2017 dataset.

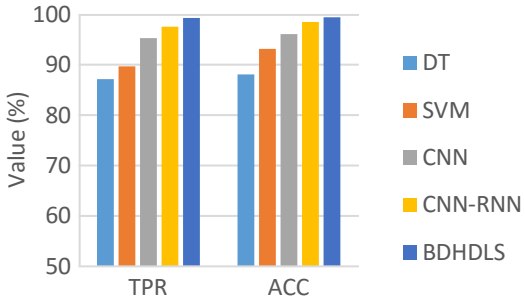

Fig. 12 TPR and ACC for CICIDS2017 dataset.

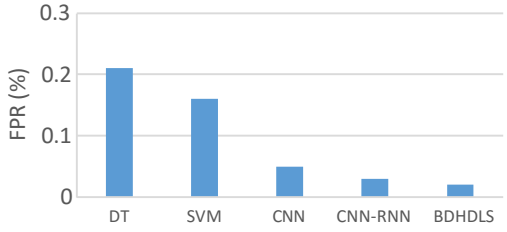

Fig. 13 FPR for CICIDS2017 dataset.

5.3.2 Statistical analysis of intrusion detection performance for the 5 × 2 CV F test in the CICIDS2017 dataset

Besides the independent testing carried out in the testing set, the combined 5 × 2 CV F test is performed in the training set to determine whether the intrusion detection performance improvement of BDHDLS over the other four models in the CICIDS2017 dataset is statistically significant. Table 5 shows "p value by F test" when binary classification of five computational models are performed in the CICIDS2017 dataset. Experimental results from Table 5 indicate that performance gains of BDHDLS over the other four computational models in terms of all evaluation metrics are statistically significant.

5.4 Results for DARPA1998 dataset

In this section, the independent test results for the DARPA1998 dataset are discussed. Figure 14 indicates that the number of samples for different intrusive attacks in the DARPA1998 dataset. U2R and R2L have the fewest number of samples. The following results are obtained from the independent test dataset. Figures 15 and 16 compare the binary classification performance of four computational models for the DARPA1998 dataset. RNN-CNN is not built for DARPA1998 dataset since the number of packets for each sample is very small. The

Table 5 "p value by F test" for binary classification in the CICIDS2017 dataset.

(%)

Model	FPR	TPR	ACC
DT	<0.1	<0.1	<0.1
SVM	<0.1	<0.1	<0.1
CNN	<0.1	<0.1	0.4
RNN-CNN	0.9	0.6	0.8
BDHDLS	N/A	N/A	N/A

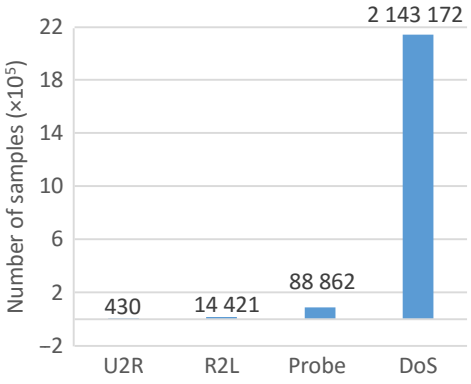

Fig. 14 Number of samples for different intrusive attacks in the DARPA1998 dataset.

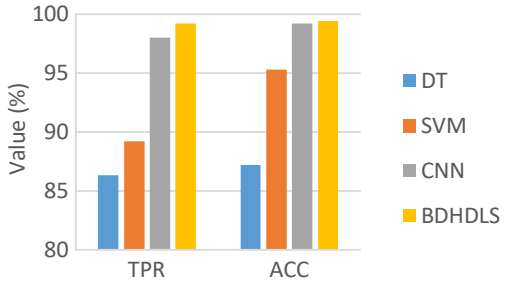

Fig. 15 TPR and ACC for DARPA1998 dataset.

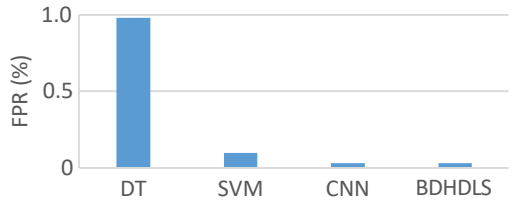

Fig. 16 FPR for the DARPA1998 dataset.

small number of packets per sample does not work well for LSTM, which need learn the temporal information of a long sequence of data. BDHDLS improves the classification performance as compared to other three computational models.

5.5 Construction time for BDHDLS when different numbers of machines are used

Figure 17 indicates the average construction time (in hours) of BDHDLS for 5 × 2 CV F test in the ISCX2012 dataset. The average construction time for BDHDLS is 12 hours when the 64 machines are used. The average construction time for BDHDLS has been reduced substantially when multiple machines are deployed.

6 Conclusion and Future Work

In this work, BDHDLS is proposed to focus its efforts on learning distinctive data distribution of specific intrusive attacks belonging to certain families. This strategy is particularly effective to capture subtle data patterns for intrusive attacks having a small number of samples.

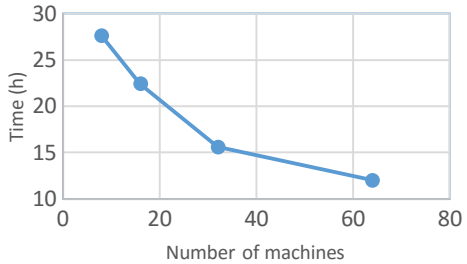

Fig. 17 Average construction time of BDHDLS for 5 × 2 CV F test in the ISCX2012 dataset.

BDHDLS also adopts both behavioral features and content features. Considering both behavioral features and content features together allows BDHDLS to analyze intrusive attack samples using both network traffic characteristics and contents in the payload. This tactics can boost the performance of IDS since previous approaches never combine both types of features together. Our study also shows that big data techniques and parallel strategies for feature selection, clustering, and training can reduce the model construction time significantly. This allows researchers to iterate faster to search the best model parameters for their computational problems.

In this work, the simple decision fusion strategy is used to combine the output of different deep learning models in the cluster. This technique may not be the optimal solution; therefore, advanced decision fusion algorithms combining outputs from different deep learning models in the tree can be experimented in the near future. We plan to experiment on using deep neural networks to merge decisions from different models in the tree instead of defining merging strategies by human experts. Since feature selection and adoption has significant impacts on performance of deep learning models, we also plan to incorporate new sets of behavioral features into deep learning models in order to enhance the performance. The fast techniques to generate multi-level cluster tree also need to be explored to further reduce the model construction. As compared to the single deep learning approach, BDHDLS uses much more computational resources in order to achieve the performance gains. How to reduce the required minimum computational resources in order to achieve the similar performance gains will be studied in the near future.

Acknowledgment

This work was partially supported by Research Initiative for Summer Engagement (RISE) from the Office of the Vice President for Research at University of South Carolina.

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. Sánchez, 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. García, 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. 1st 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 Politècnica 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. 4th 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.

Wei Zhong received the PhD degree in computer science from Georgia State University, USA, in 2006. He is a full professor in the Division of Math and Computer Science, University of South Carolina Upstate. He is an elected fellow of International Society of Intelligent Biological Medicine. He is also the IEEE senior member. His research interests include deep learning, data mining, bioinformatics, and high performance computing.

Ning Yu currently is an assistant professor at the State University of New York College at Brockport, USA. He earned the PhD degree in computer science from Georgia State University in 2016 and has published more than 20 papers in prestigious journals, such as *IEEE Transactions*, *BMC Bioinformatics*, and *PLOS One*. His current research focuses on big data analytics, deep learning, network and information security, information processing, and high performance computing.

Chunyu Ai received the BS and MS degrees in computer science from Heilongjiang University, China in 2001 and 2004, respectively, and the PhD degree in computer science from Georgia State University, USA in 2010. She is currently an associate professor in the Division of Math and Computer Science, University of South Carolina Upstate. Her research interests include wireless sensor networks, data management, machine learning, and social networks.