Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
() : –
© TÜBİTAK
doi:10.3906/elk-2102-89

A new distributed anomaly detection approach for log IDS management based on deep learning

Murat KOCA[1,2], **Muhammed Ali AYDIN**[2,*], **Ahmet SERTBAŞ**[2], **Abdül Halim ZAİM**[3]
[1]Provincial Directorate, Republic of Turkey Ministry of Industry and Technology, Hakkari, Turkey
[2]Department of Computer Engineering, İstanbul University-Cerrahpaşa, İstanbul, Turkey
[3]Department of Computer Engineering, İstanbul Commerce University, İstanbul, Turkey

Received: 18.02.2021 • **Accepted/Published Online:** 18.04.2021 • **Final Version:** ..2021

Abstract: Today, with the rapid increase of data, the security of big data has become more important than ever for managers. However, traditional infrastructure systems cannot cope with increasingly big data that is created like an avalanche. In addition, as the existing database systems increase licensing costs per transaction, organizations using information technologies are shifting to free and open source solutions. For this reason, we propose an anomaly attack detection model on Apache Hadoop distributed file system (HDFS), which stands out in open source big data analytics, and Apache Spark, which stands out with its speed performance in analysis to reduce the costs of organizations. This model consists of four stages: the first of which is to store instant data on HDFS in a distributed manner. In the second stage, the log data generated in the network traffic are analyzed by taking the data on Apache Spark and including the log data created by HDFS. In the third stage, the data preprocessing stage and with the CUDA parallel programming in the TensorFlow library, we apply our deep learning (cuDNN) method to the distributed anomaly detection with the computational support of GPUs. In the last stage, the generated alarms are recorded on HDFS again. We conducted comparative experiments with the approach we propose to detect cyberattack anomalies in log data management with the classification methods used in machine learning. The results obtained in these experiments appear to provide a promising gain in performance evaluation metrics compared to the other available methods.

Key words: Big Data, Deep Learning, Cyber Security, Log IDS, Spark, CUDA

1. Introduction

Rapid development of technology brought along many new attack types. It also enabled existing attacks to be differentiated by gaining new features. This situation has caused material and moral loss to a considerable number of institutions, organizations and companies. The data generated by recording the events is called a ”log”. Logs, depending on the environment in which the events take place, are divided into many types; software logs, operating system logs, network server logs, etc.

The advancing technology has unfortunately brought along many risks as well as numerous possibilities and opportunities. This situation has increased the amount of available data exponentially rather than linearly, so the storage of data has become a separate problem and has increased the costs. Therefore, the issue of network security, which allows the protection of data privacy, integrity and accessibility, has gained great importance and has taken a critical position. Any behavior that threatens the privacy, integrity or accessibility of data is called cyber-attack. Multiple types of attacks exist on the network, the highlights: SYN attack (SYN flooding),

[*]Correspondence: aydinali@istanbul.edu.tr

DoS/DDoS attacks, packet sniffing, spoofing, viruses, spyware, DNS over HTTPS (DoH), web attacks and darknet [1, 2].

Intrusion is the act of violating the cyber-security policy or legal protections of an information system. As defense devices against attacks here; intrusion detection systems (IDS), intrusion prevention systems (IPS) and firewalls. Most of the time, they can be used with this system under the name of intrusion detection and prevention systems. A firewall is a combination of software and hardware that protects the network from external attacks. The firewall is considered to be the primary defense border in protecting data from attacks from outside.

Nowadays, it has to store and manage petabytes of data that grows like an avalanche. Traditional existing infrastructure is lacking in big data processing. In addition, existing database systems provide per transaction licensing. Today, it is classified as big data due to the qualities (volume, variety, velocity, value, veracity and variability) of the network traffic generated in corporate environments [3]. Analyses of these data have made it necessary to use big data tools.

Apache Hadoop is an open source framework for computing due to its distributed structure and scalability, because it is both open source and reliable[1]. Hadoop distributed file system (HDFS) is a distributed file systems designed to work with commodity hardware systems and to minimize error tolerance. Thanks to these features, it is a distributed file system that can store data on thousands of servers and data mapping/downloading works that share jobs between these machines. This storage system has a master/slave architecture. Big data is divided into chunks managed through different nodes in the Hadoop cluster [4].

Hadoop is an open source library written in Scala that allows us to treat big data as distributed. Generally, Spark can be considered as MapReduce equivalent. Storing data in HDFS, after the data can be processed simpler and faster on Apache Spark[2]. It is understood that it is faster than MapReduce and that Spark is 100 times more efficient in memory operations than MapReduce according to the official web page [4, 5].

For the above reasons, we keep our data on Apache Hadoop HDFS, and stand out with its performance in analysis and simplicity in coding, and in the analyzes we coded with pyspark on Apache Spark, we first implemented our data recognition and several data preprocessing methods. Following these processes, it is the stage of creating a deep learning model using distributed Keras library, which performs best within the model and data parallelization. Keras library is written in Python programming language and allows more comfortable coding with APIs of TensorFlow, CNTK or Theano[3]. TensorFlow in the background that will create an easily distributed deep learning network that can integrate with Spark with the PySpark pipeline line[4]. With the CUDA parallel programming in the TensorFlow library, we apply our deep learning (cuDNN) method approach to the distributed anomaly detection with the computational support of GPUs [6]. Unlike of the studies we have researched, our main contribution is keeping the performance and accuracy rate of GPUs computational support by implementing cuDNN with the new algorithms we have developed for generating all alarms of these algorithms are executed on Apache Spark.

In the second part of this article, brief information was given about the studies related to our work in this field in Section 2, the architecture of the proposed model is explained in Section 3, the results of the analyses

[1]Securosis LL (2012). Securing Big Data: Security Recommendations for Hadoop and NoSQL Environments [online]. Website https://cdn.securosis.com/assets/library/reports/SecuringBigData_FINAL.pdf [accessed 8 February 2021].

[2]Apache Spark Foundation (2018). Apache Spark 2.3.1 Documentation [online]. Website: https://spark.apache.org/docs/2.3.1/ [accessed 31 December 2020].

[3]Keras.io (2019). Home-Keras Documentation [online]. Website: https://keras.io/ [accessed 31 December 2020].

[4]TensorFlow (2018). TensorFlow Documentation [online]. Website: https://www.tensorflow.org/ [accessed 31 December 2020].

and discussions are described in Section 4. Finally, the result of this study is indicated in Section 5.

2. Related work

When the studies in the literature are examined, it can be seen that there are various definitions of intrusion detection systems. Attack detection systems: It is a process that analyzes behaviors that threaten the security of a system to destroy, capture, confidentiality, and integrity of data on any network or device [7]. Besides various definitions, there are IDS implemented with multiple machine learning algorithms.

Tan et al. designed a system based on the analysis of automated system logs in distributed systems called SALSA. This system is used to examine logs to monitor the flow of control and data in a distributed system and reproduce state machine-similar views of system execution at each node. SALSA's shortcomings are that it is not fast and analysis cannot be made about the instant status of the data. By default, Hadoop's log4j configuration creates a separate log for each of the logs, and each log is stored on the executed local file system and analyzed [8].

Xu et al. approach for automatic detection and monitoring of abnormal behavior in console logs using data mining and statistical learning methods. There is a two-stage structure in this system. In the first stage, the determination of the common pattern and the combination of distributed prediction methods are used to capture the dominant pattern. In the next stage, principal component analysis, which has anomaly detection features, are used to determine the main problem [9].

Lee and Lee proposed a study that offers an approach with Hadoop components to measure and analyze Internet traffic. In this experiment, they recorded throughput rates of up to 14 Gbps in some trials for DDOS detection and used 30 or more nodes in each data set. However, for some types of analysis, they observed that their slower results approached 6 Gbps. While performing 5 different analyzes, they also tested various options such as changing the number of cluster nodes. It has been observed that their studies also consider real-time traffic monitoring methods, not only previously recorded traffic data [10].

Suthaharan proposes a large data model to detect attacks. This model captures network traffic and stores this data to the cloud storage system or locally to HDFS. The data can be taken from the places where they are saved and used again [11].

In the study by Desai and Gaikwad; a hybrid STS has been developed to detect both the insider attacks and the external attacks. This study performs signature matching for the detection of insider attacks while detecting external attacks with fuzzy genetic algorithms. The presented system can work online as well as offline. There is not enough information about the data set used in the work and the resulting chart consists of 50 records in total. In addition, the success rate of the study was not given and it was claimed that it was successful from some systems [12].

Marir et al. suggests a distributed approach for detecting abnormal behavior in large networks. With this model, they developed a system that detects abnormal behavior from distributed large-scale network traffic data by combining support vector machines with deep learning methods [13].

Karataş et al. investigated a comparative study of the literature with a deep learning-based intrusion detection system approach and researched intrusion detection systems. In addition, they made a research about the available data sets in their work [14].

Unlike the above studies, the main contribution to our work is primarily performed on the distributed Apache Spark cluster with multinode.Then, in the most important stage we attach importance to is the data preliminary stage. Here we extract and standardize the insignificant data. We aim to maximize the performance of network traffic attack detection by using cuDNN thanks to the TensorFlow library in order to benefit from the parallelization on the GPU card. The performance and accuracy results from we have obtained as shown in Table 4. We train what we have gained from data preparation with the Keras distributed deep learning model and generate alarms.

3. Architecture of the proposed model

Existing traditional infrastructure falls short of computing big data and generating analytical new data on them. In addition, we aim to solve these problems by using the components in the Apache Hadoop ecosystem, which is open-source software and is used by many technology giants today, due to cost-increasing reasons such as per-transaction licensing in existing database systems.

Our proposed model is shown in Figure 1, we first collect the data of network traffic, e-mail, and internal traffic events in the HDFS. After this step, we pull the logs on HDFS onto Apache Spark with the data collected here to create important alarms. We process our data in two stages on Apache Spark. In these phases, after passing our data through the preliminary stage, we generate attack alarms that occur with the huge distributed deep learning method. Finally, we record the alarms that occur here on HDFS again.

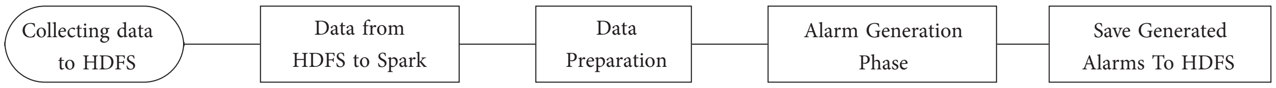


Figure 1. Architecture of the proposed model.

3.1. Collection of data from network traffic

In the first stage of the proposed model, we collect traffic records on the external network and the internal network. We save these collected data in the distributed architecture Hadoop distributed file system. In addition to these records, we also upload the log records created by Hadoop into the system. Our aim here is to track system administrators and administrators with the necessary permissions. With the Zeek[5] tool, the process of capturing network traffic has been automated. The traffic recording features we collect here are as follows:

- SYSLOG: It has been developed using TCP / IP protocols in which log records on the system are kept, and stands out with their value to system management. It uses the user datagram protocol (UDP) as port 514 as the underlying transport layer mechanism [15].

- SNMP (simple network management protocol): It is the application layer protocol that provides the most important contributions to the network system administrator. It is used to facilitate the management of devices in large networks. In this technology, we can learn the basic information of router, switch, access server, bridge, and other devices in the network [16].

- NETFLOW: Provides data about users and routing traffic on the network. Being template-based allows users to follow new developments as they can redesign according to their strategy. In a NetFlow stream; Records are UDP/TCP source and destination ports, IP source and destination addresses, ICMP type and code, Input frames, etc. [17].

[5]Zeek (2018). Zeek documentation [online]. Website: https://docs.zeek.org/ [accessed 25 March 2021].

- EVENT LOG: All transactions performed by all users and applications on the system are recorded [18]. They are vital records for cyber incident response teams (CIRT). The cybersecurity incident is recorded with the time tags of the users.

3.1.1. Recording of data in HDFS system

Thanks to the distributed storage service and its ability to reproduce bytes of network data in case of hardware malfunction at any time, reliability is increased [19]. Allows storage that separates data that occurs at a large data scale into scalable metadata. It stores very large files, usually larger than 1 terabyte each, allowing it to be moved to different media for analytical reviews. For these reasons, we save our collected data in the Hadoop file system with at least 3 replications.

3.1.2. Moving data on Apache Spark and data preliminary preparation

Data is extracted from the network, servers and system administrators by collecting the network traffic on HDFS and adding SparkSession from the Python Spark library to Apache Spark from HDFS[2]. In the next stage, the data forward processing stage was started. Getting to know our data is the most important step in big data analysis. For this, the Pyspark library is very rich. The path we use here is shown in Figure 2.

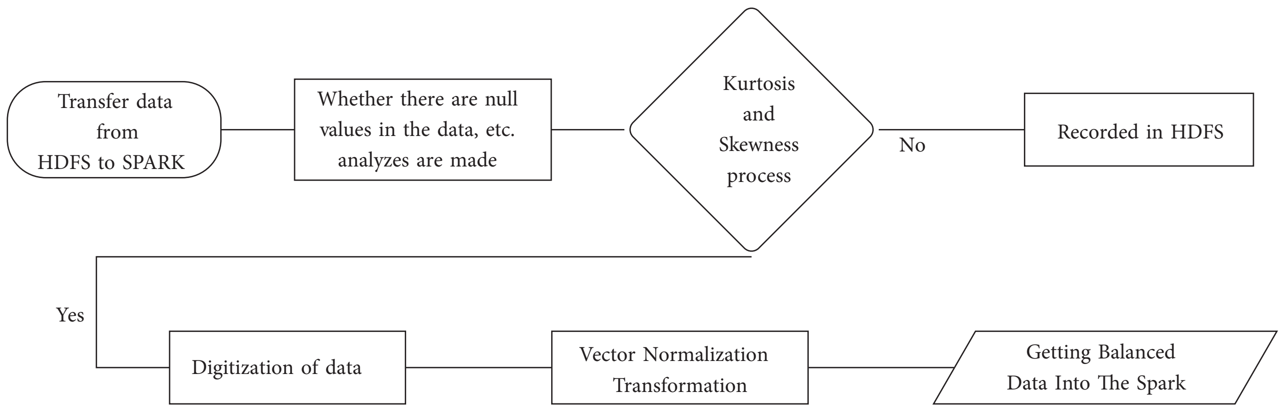

Transfer data from HDFS to SPARK
Whether there are null values in the data, etc. analyzes are made
Kurtosis and Skewness process
No
Recorded in HDFS
Yes
Digitization of data
Vector Normalization Transformation
Getting Balanced Data Into The Spark

Figure 2. Data preprocessing flowchart.

The most important thing in data preprocessing is that it has made learning processes and establishing connections difficult during the deep learning phase due to reasons such as the value of the data set being higher than –2 to 2. As a result, it has a great effect on accuracy and positive-false situations in training data. For this reason, we apply Algorithm 1 to our data at this stage.

Algorithm 1 Data preparation phase balancing between –2 and 2.

```
1: $i \leftarrow 0$
2: for ( i < feature_lists) do
3:     if (The values in the feature list are in the range of --2 to 2) then
4:         We process the selected data through Kurtosis and Skewness processes.
5:     end if
6: end for
```

With Algorithm 1, when we first consider the normal distribution as a bell curve, it shows whether the peak of the bell is skewed to the right or the left. With the Kurtosis we apply, we obtain data on the vertical plane of the bell, and with the skewness function, on the horizontal plane of the bell[20]. Skewness and kurtosis values between −1.5 to 1.5 or between −2 to 2 are considered normal [21]. The values selected here are vectorized using the VectorAssembler function, as shown in Algorithm 3, and then balanced with the StandardScaler function[2]. With StandardScaler, it standardizes the means of the data to be 0 and the standard deviation to be 1. The main reason we chose this method is that it is useful for data with negative values.

3.1.3. Distributed anomaly detection approach for log IDS alarm generation phase based on deep learning

In order to make the necessary analysis of the data received from the Spark Pipeline[2] after preprocessing the data, the flow stages shown in Figure 3 are in the log IDS alarm generation phase; first of all, we randomly sort our training and test data.

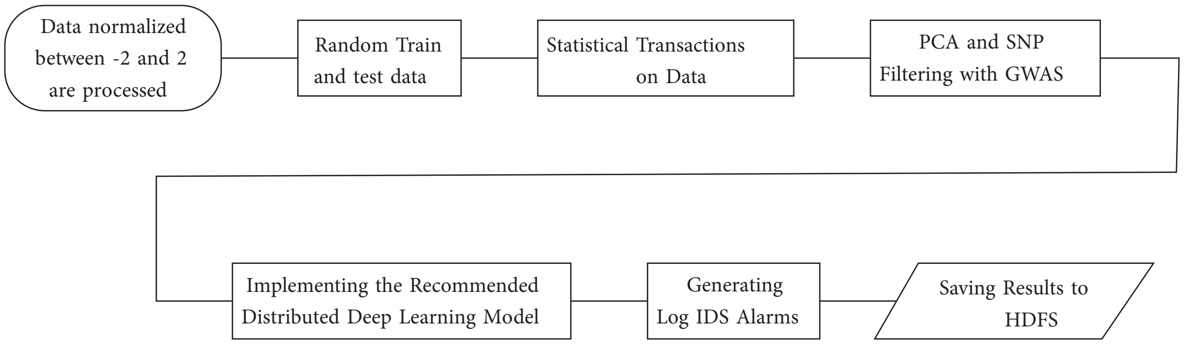

Data normalized between -2 and 2 are processed
Random Train and test data
Statistical Transactions on Data
PCA and SNP Filtering with GWAS
Implementing the Recommended Distributed Deep Learning Model
Generating Log IDS Alarms
Saving Results to HDFS

Figure 3. Distributed anomaly detection approach for log IDS alarm generation phase based on deep learning flow diagram.

In the light of this information, as shown in Algorithm 2, principal component analysis (PCA) basically shows a multidimensional data with fewer variables by capturing the basic features of the data [22]. Here, with the training data set, which is important for us, we determine the areas where our test data set is concentrated or completed. In the next step, genome-wide association studies (GWAS), the statistical method shown in Algorithm 2, is the exploration studies of different genetic characteristics of the target group by making target and control comparisons of groups with more than one sample. We determine the significance of the correlation between the training and the test data, the link between the source and the target by estimating the SNPs missing the number of SNPs in the linkage disequilibrium (LD) panels in our PCA results.

Algorithm 2 Elimination of deficiencies and imbalances in training and test data.

```
$X = Unlabelled\_Data, Y = Labelled\_Data$
Data Set Divided (X, Y, Random Size, Test Size)
$i \leftarrow size\_to\_reduce$
for ( i < feature_lists) do
   PCA function is applied
   for ( i < feature_lists) do
      linregress(feature_list [b], y_train) GWAS link imbalances balancing with function
   end for
end for
```

We will use deep learning, a subdivision of artificial intelligence, to minimize the errors arising from manual extraction of the features in the anomaly detection we will use and benefit from its proven performance. In the architecture we use in deep learning, we model the input layer, n hidden layer and output layers. Machine learning is about being able to identify the properties of a data set and apply them to new data.

Then we create a distributed deep learning model. Here, using the model = Sequential () feature of Keras [23] in the background and these processes are explained in Algorithm 3, it is easy to simply add layers and create a deep learning model with all desired settings (unit count, dropout, arrangement-l2, activation functions, etc.). Thanks to the CUDA parallel computing power on the computing support of GPUs, allowing the use of the TensorFlow library. By combining CUDA and deep learning using the TensorFlow library, we provide the power of distributed parallel computing with cuDNN support [6]. We apply different optimizations and cross entropy to separate the results into labeled data.

In our distributed deep learning model, we used a fully connected network structure. In this structure, the number of neurons in each layer, the number of epoch, learning speed, etc., all metaparameters are automatically determined according to their attributes in the data sets and Section 4 is explained in detail as an example. This type of network architecture is called multilayer perceptron (MLP)[24]. When placing layers, we equalize the input size of each layer (input_dim) to the output size of the previous layer, but here we apply dropout to prevent memorization and overfitting in some layer outputs.

We will also use the Elephas library here. Elephas aims to maintain the simplicity and top availability of Keras, because of allowing rapid archetype of distributed deep learning models that can be set to work on large data sets[6]. In the artificial neural network we use in the Keras library with Elephas, we also balance the dropout as 0.3 to optimize the learning rate as 0.01 and prevent the memorization of our network. We used the rectified linear unit (ReLU) function because the activation functions we will use here are less than the sigmoid and hyperbolic tangent activation functions that take value in the range of (0, + infinite) in hidden layers [25]. In the last output layer, we produce outputs with the softmax activation function that interprets the given input in the hidden layers by generating values in the range of 0–1 belonging to a certain class [23]. We make certain statistical analyzes for log IDS alarms generated here.

3.1.4. Save generated logs IDS alarms to HDFS

With the log IDS alarms generated, we analyze the statistical accuracy measurements, probability metrics, regression metrics and classification measurements based on true / false positives and negatives, and record all results on the HDFS system. The statistical measurements here will be examined in detail in Section 4.

4. Analysis and experimental results

In this section, we will test our proposed approach and discuss its results. First, we discuss the data sets we use, then our evaluation criteria, and finally our comparative results. We tested our work on publicly available data sets that stand out in the literature until today. According to our research, intrusion detection systems have created a large number of data sets since 1998. We have worked on the data that stands out in studies from that date to the present.

[6]Maxpumperla/elephas (2005). Distributed Deep learning with Keras & Spark [online]. Website: https://github.com/maxpumperla/elephas [accessed 28 January 2021].

Algorithm 3 Proposed log IDS anomaly detection structure with distributed deep learning model on Apache Spark.

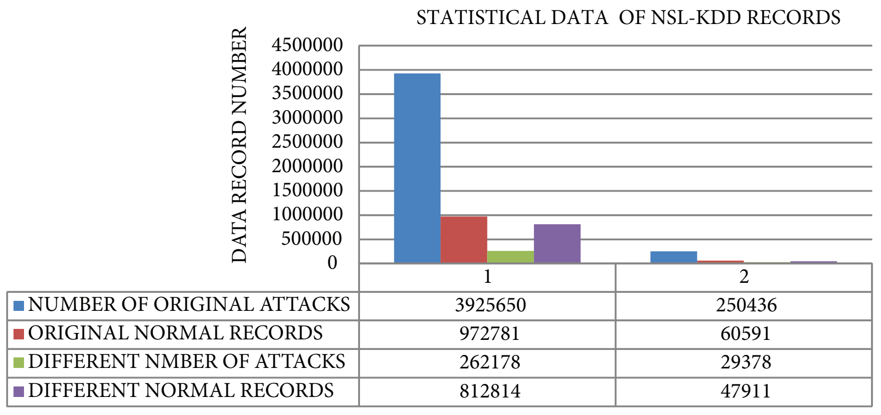
```
Input:We take our DataSet into Apache Spark Pipeline (Label Index, scaled features)
Output:Pipeline Stages [StringIndexer, VectorAssembler, StandardScaler]
$i \leftarrow 0$
while $i > 10$ do
   Making adjustments for the TensorFlow parallelization process using Sequential () as the model from the Keras library.
   if ( i <= n_$epochs$) then
      Getting Training Sets (train_data, test_data), Hyper parameters (Classification numbers, data attributes to be trained, dropout%, regularization - learning rate, activation functions) are set. The data set on the pipeline is trained with the deep learning model and the network is trained
   end if
end while
```

4.1. Data sets

We discussed the data sets we use, the traffic generated on the external network and the internal network, and the data sets currently used from the event records on the Hadoop. We chose these data sets from among those that stand out in the citation ranking today. The cybersecurity datasets that we will test the effectiveness of the approach we propose is described below.

4.1.1. NLS-KDD

KDD CUP'99 is generally a synthetic data structure obtained from DARPA'98. The synthesized data does not resemble the traffic on real networks. It has empty packets. There is no clear definition of the attacts. Therefore, there is a need for a new network attack detection system data set. Therefore, Tavallaee et al. recommended NSL-KDD, a 10 percentage reduced and destabilized version of the KDD CUP'99 dataset in 2009 [26]. They fixed the disadvantageous properties of 10 percent of KDD CUP'99 in two phases. Firstly, they eliminated unnecessary data from both training and test sets as shown in Figure 4.



	1	2
🟦 NUMBER OF ORIGINAL ATTACKS	3925650	250436
🟥 ORIGINAL NORMAL RECORDS	972781	60591
🟩 DIFFERENT NMBER OF ATTACKS	262178	29378
🟪 DIFFERENT NORMAL RECORDS	812814	47911

Figure 4. NSL-KDD statistical data records.

In the next stage, they broke it down into levels of difficulty to increase data diversity. After this process, they chose records from each level in inverse proportion with the percentage downloaded 10 percentage. In this way, NSL-KDD has been modified in a way that it has reasonably balanced quantity of records in both training and test data [7].

4.1.2. CIC-IDS2017

The Canadian Institute of Cyber Security Intrusion Detection System (CIC-IDS 2017) proposed a public anomaly-based network intrusion detection system data set in 2017[8]. This data set is to provide a reliable and up-to-date data set. Moreover, the authors claim that this dataset is free of all the weaknesses of other existing network intrusion detection system datasets. There is a network structure with a framework consisting of two separate networks, the attack groups and the groups exposed to this attack. The attacked group is called the B-Profile, it is used to ensure benign behavior [27]. The other attack group is used to generate attack traffic on the network. Both groups are equipped with the needed network devices and computers running different operating systems. They also used the CICFlowMeter Analyzer to analyze the PCAP data, whose statistical properties are described in Table 1, for five business days. The network traffic generated uses HTTP, HTTPS, FTP, SSH, and e-mail protocols.

Table 1. CIC-IDS 2017 statistical data of records.

Event/day	Monday	Tuesday	Wednesday	Thursday	Friday
Attack type	Normal activity	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity
Attack desc.	Benign	Brute force	DoS/ DDoS,	Web attack – brute force, SQL injection, infiltration,	Botnet
Size (GB)	11	11	13	7,8	8,3
Victim description	-	Webserver, Ubuntu	Webserver, Ubuntu	Webserver, Ubuntu, Windows Vista, MAC,	Win 10, Ubuntu16
Attacker description	Normal	Kali	Kali	Kali, Vista	Kali, Three Win 8.1

4.1.3. CSE-CIC-IDS2018

The main purpose of this data set[9] is to develop a systematic approach to create a diverse and comprehensive benchmark data set for intrusion detection based on the creation of user profiles containing abstract representations of events and behaviors seen in the network. The profiles will be combined to create several sets of data covering part of the evaluation area, each with a unique set of features. As can be seen in Table 2, this cluster consists of seven different attack types: DoS, DDoS, brute force, infiltration, Botnet, Web attacks, and Heartbleed. The attack infrastructure consists of 50 machines, 5 victim groups, 420 machines, and 30 servers. The dataset captures the network traffic and system logs of each machine with 80 attributes derived from the network traffic captured using CICFlowMeter-V3 [28].

[7]Canadian Institute for Cybersecurity (2005). NSL-KDD Datasets [online]. Website: https://www.unb.ca/cic/datasets/nsl.html [accessed 04 January 2021].

[8]Canadian Institute for Cybersecurity (2017). CIC-IDS 2017 Datasets [online]. Website: https://www.unb.ca/cic/datasets/ids-2017.html [accessed 04 January 2021].

[9]Canadian Institute for Cybersecurity (2018). CSE-CIC-IDS2018 Datasets [online]. Website: https://www.unb.ca/cic/datasets/ids-2018.html [accessed 04 January 2021].

Table 2. CSE-CIC-IDS 2018 statistical data of records.

Duration	One day	One day	One day	Two days	One day	Two days	One day
Attack type	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity	Attacks + normal activity
Attack desc.	Brute force attack	DoS	DoS	Web attack	Botnet attack	Web attack	DoS
Tools	FTP and SSH Patator	Hulk, Goldeneye, Slow Loris, Slow http test	Heart leech	Web App (DVWA), (XSS and brute force)	Ares, screenshots and key logging	Web App (DVWA), (XSS and brute force)	Heart leech
Victim desc.	Ubuntu 16.4 (Web Server)	Ubuntu 16.4 (Apache)	Ubuntu 12.04 (Open SSL)	Ubuntu 16.4 (Web Server)	Windows Vista, 7, 8.1, 10	Ubuntu 16.4 (Web Server)	Ubuntu 12.04 (Open SSL)
Attacker desc.	Kali	Kali	Kali	Kali	Kali	Kali	Kali

4.2. Evaluation criteria

We will introduce the statistical methods of the approach we propose, accuracy criteria, and classification measures based on True / False positives and negatives.

4.2.1. Accuracy measurements

In this measurement, it creates the local variable that is used to calculate the frequency at which the predictions match the correct values. The frequency ranges we will use vary according to the variables. First of all, it shows our error rate in our confusion matrix evaluation according to true and false values. The confusion matrix is explained as follows and equations to be used to evaluate our approach in Table 3.

- TN(true negative): Indicates that the actual data and the predictions are the same, that is, there is no attack.
- TP(true positive): Indicates that the actual data and the predictions are the same, that is, the attack has occurred.
- FN(false negative): Indicates that the actual data and the predictions are different, that is, the actual data include attack but the predictions are normal.
- FP(false positive): Indicates that the actual data and the predictions are different, that is, the predictions indicate attack but the actual data show no attack.

4.3. Evaluation results and analysis of the proposed model

In this section, the model we propose consists of two stages: data preliminary stage and data training. During the data preparation phase, we first considered our 3 main data sets for the recognition phase. At this stage, we have determined the data attributes that stand out in the attack categories of our data and eliminated

Table 3. Criteria and equations to be used to evaluate our approach.

Evaluation criteria name	Evaluation criteria description	Equation
Accuracy	It is calculated as the ratio of correctly predicted features to the whole data set. This evaluation criterion shows the performance of the model.	$\frac{TP+TN}{TP+TN+FP+FN}$
Sensitivity, recall, hit rate, or true positive rate (TPR)	This criterion is a measure of how well you know the actual attacks.	$\frac{TP}{TP+FN}$
Specificity, selectivity or true negative rate (TNR)	It measures the rate of normal/harmless events correctly identified.	$\frac{TN}{TN+FP}$
Precision or positive predictive value (PPV)	It shows how many of the values predicted as an the attack.	$\frac{TP}{TP+FP}$
Negative predictive value (NPV):	It shows how many of the values predicted as nonattack does not attack.	$\frac{TN}{TN+FN}$
Fall-out or false positive rate (FPR)	It is calculated as the ratio between the number of false predictions attack events and the total number of predicted attack events (regardless of classification).	$\frac{FP}{TN+FP}$
F1 score	It is the harmonic average made with the results of precision and recall. The reason for the harmonic mean instead of weighing is that contradictory situations should not be taken into account.	$\frac{2*TP}{TN+\frac{1}{2}(FP+FN)}$
AUC(area under the ROC curve)	ROC is a probability curve applied to different classes. There are ROC false positive rate (FPR) and true positive rate (TPR). The more the area under this curve, the more machine learning models are accepted as successful in distinguishing the given classes	$\int_x^1 TPR(FPR^{-1}(x))\,dx$

the unnecessary attributes in the data. Here we focused on the standardization of data that affects big data training performance. After converting the attributes into the standard described in Algorithm 3 into vectors using the VectorAssembler function, we balanced them with the StandardScaler function. In the next stage after balancing the data, we use a deep neural network structure to detect outliers. After fine-tuning the network we have created, we determine our model.

In the Apache Spark machine learning library, we evaluate our approach we proposed with five different classification methods: support vector machine (SVM) [29], restricted Boltzmann machines (RBM) [30], deep belief network (DBN) [30], multilayer perceptron (MLP) [24] and distributed multilayer ensemble SVM (MLE-SVM) [13]. The results here are shown in Table 4. For the NSL-KDD dataset, there are 2 classes and 24 data attributes, the dropout is 30%, the regularization-learning rate is 0.01 and the number of epoch is set to 50. For the training, there are 24 input nodes, 2 hidden layers, and 2 output nodes. For the remaining data set, there are 2 classes and 80 data attributes, the dropout is set 50%, the regularization-learning rate is 0.03, and the number of epoch is set to 250. For the training, there are 80 input nodes, 4 hidden layers, and 2 output nodes. For the all datasets, while the ReLu activation function is used in hidden layers, softmax activation function is used for output layer. In these networks, we used "Adam" as optimizer and "categorical_crossentropy" as loss [23] . We compared the performance of the proposed model with that of related work using training time, testing time, F-1 score, recall, precision, and area under ROC metrics, as shown in Table 4.

Table 4. Comparison of the proposed model with the related work.

Datasets	Evaluation metrics	Classifiers					
		SVM [29]	RBM [30]	DBN [30]	MLP [24]	MLE-SVM [13]	Proposed approach
NSL-KDD	Train T.	25.62	21.37	21.42	23.15	20.12	19.47
	Test T.	0.21	0.15	0.16	0.18	0.17	0.14
	F-1 score	0.9012	0.9475	0.9417	0.9118	0.9398	0.9617
	Recall	0.8813	0.9076	0.9116	0.9237	0.9297	0.9324
	Precision	0.9312	0.9572	0.9692	0.9367	0.9640	0.9715
	AUC-ROC	0.9398	0.9614	0.9762	0.9419	0.9795	0.9876
NSL-KDD	Train T.	92.17	77.65	78.15	78.87	79.78	75.38
	Test T.	0.82	0.70	0.71	0.91	0.76	0.68
	F-1 score	0.9217	0.9301	0.9265	0.9281	0.9583	0.9871
	Recall	0.9613	0.9615	0.9717	0.9386	0.9387	0.9864
	Precision	0.8916	0.9508	0.9456	0.9143	0.9265	0.9852
	AUC-ROC	0.9417	0.9750	0.9684	0.9574	0.9790	0.9896
CSE-CIC-IDS2018	Train T.	97.17	85.65	80.17	80.87	89.28	78.78
	Test T.	0.98	0.83	0.86	0.95	0.84	0.82
	F-1 score	0.9117	0.9462	0.9301	0.9118	0.9672	0.9751
	Recall	0.9428	0.9578	0.9412	0.9278	0.9423	0.9654
	Precision	0.9027	0.9532	0.9412	0.9399	0.9310	0.9892
	AUC-ROC	0.9545	0.9601	0.9715	0.9671	0.9810	0.9845

In the multiclass evaluation metrics we have used, we have dealt with the confusion matrix. Here, they should evaluate the concept of positive and negative with predictions and labels, taking the value of more than one attack class, correct for the same attack groups, and incorrectly for all other classes. Thus, a true positive occurs when the guess and tag match, while a true negative occurs when neither the guess nor the tag takes the value of a particular class. The precision rate ratios of the approach we have obtained using confusion matrix have been achieved as 0.9715 in NSL-KDD, 0.9896 in CICIDS2017, and 0.9845 in CSE-CIC-IDS2018. Its performance with other classifications has been evaluated as more successful as shown in Figure 5. The performance of the precision value here indicates that the false positive ratio is low compared to other classifications. Thanks to Algorithm 2 in the model we propose, we achieve this performance.

In our evaluation with binary classifiers, we considered it as a special case of multi-class classification by using certain cyber-attack cluster elements to separate them from groups that generate possible alarms into normal and attack groups. We evaluated the area under receiver (AUC) operating characteristic (ROC) as our most important performance measure here. This criterion is the level we want the success of our model with the convergence of the TPR value to 1, but we accept that the performance of our model is weak if the ratio of FPR is high when the TPR converges to 1. The greater the area under the ROC curve, the greater the success rate of our model, by concluding that our AUC under ROC value is close to 1. As shown in Figure 6, the success of our approach in the validation test we made with our data sets was determined as to be 0.9876 in NSL-KDD, 0.9574 in CICIDS2017, and 0.9671 in CSE-CIC-IDS2018. It is seen that this is a success when compared with other evaluation criteria.

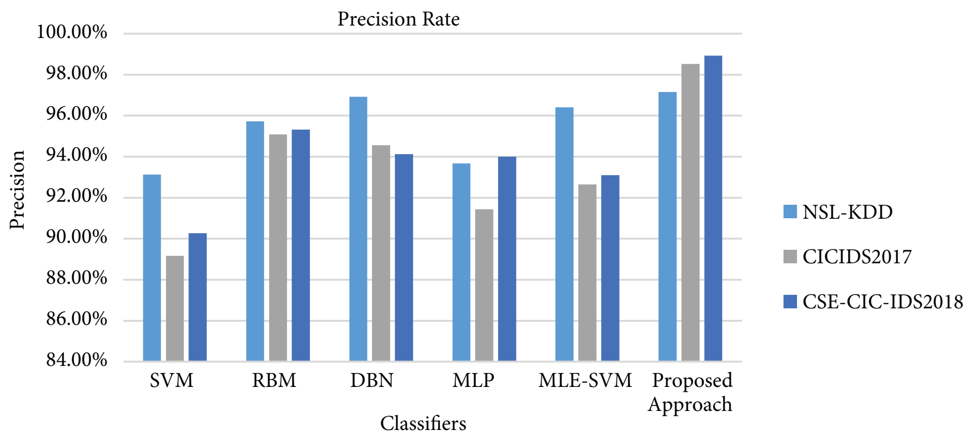


Figure 5. Precision rate percentage scores of classifiers used in data sets.

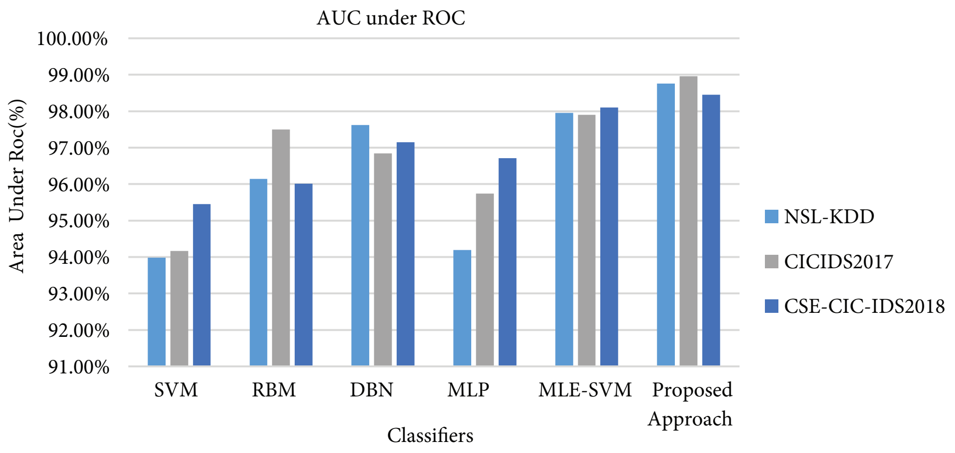


Figure 6. AUC under ROC benchmark comparison.

We performed all experiments in a virtual Apache Hadoop cluster with 6-node (1-master, 5-slave). In this structure, we installed the master node on Intel (R) Xeon (R) Silver 4110 CPU @ 2.10GHz and 16 GB RAM, and each slave nodes are Intel (R) Xeon (R) Silver 4110 CPU @ 2.10GHz and 8 GB RAM. We used Linux (Ubuntu 18.04.5 LTS) as the operating system. We set Apache Spark version 2.3.1, Hadoop 3.2.1 version, TensorFlow version 2.4.0. As GPU, we used CUDA 11.0 and cuDNN SDK 8.0.4 version on Nvidia GeForce GTX 1050 Ti board. We propose that 19.47s in NSL-KDD and 75.38s in CICIDS2017, and 78.78s in CSE-CIC-IDS2018 recorded in the Figure 7 over time. Compared to other verification classifications, it is seen that its performance is faster, as shown in Figure 7. We evaluate our performance thanks to the CUDA parallel computing power on the computing support of GPUs, allowing the use of the TensorFlow library.

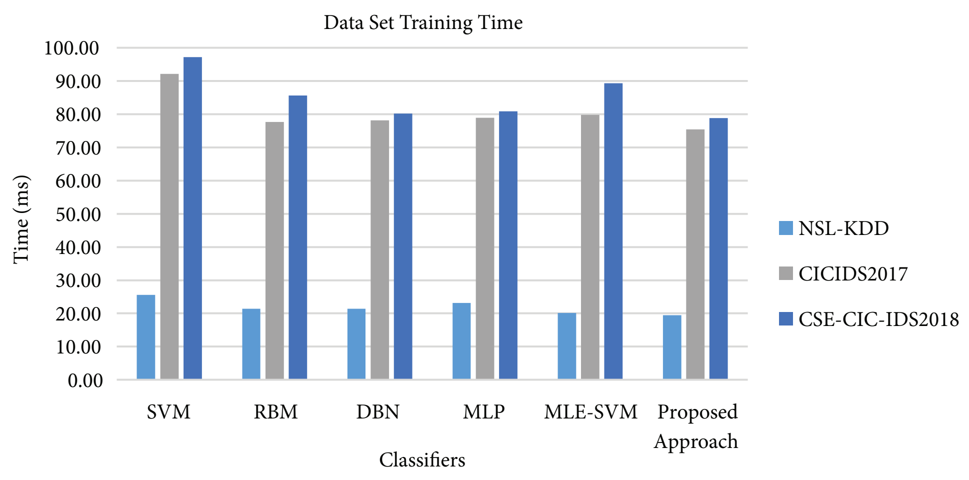


Figure 7. Training time comparison of data sets according to classifiers.

5. Conclusion

In this article, an approach is proposed for the fast and efficient cyber attack detection system of big data log management, which has become very important for system administrators today. The performance of the proposed approach has been evaluated using large log data using the power of parallelizing TensorFlow with the Keras library on Apache Spark. We used NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018 real-time datasets for performance evaluation of the proposed approach using different classification models.

In comparative machine learning methods, it has been observed that our large distributed deep learning model, which stands out with its performance in artificial neural networks today, is more successful than F-1 score, recall, precision, and area under ROC evaluation metrics. The main reason for this success is the balancing and data extraction power of deep learning that we have done in data preparation. In addition, the time spent on data training is due to Apache Spark's machine learning capability and our distributed deep learning model. In future studies, we will aim to investigate the intrusion detection system on identity privacy.

Acknowledgment

This work is a part of the Ph.D. thesis titled "Design of an intrusion detection system against cyber attacks in big data logs management" at the Institute of Graduate Studies, İstanbul University-Cerrahpasa, İstanbul, Turkey.

References

[1] Baykara M, Das R. A novel hybrid approach for detection of web-based attacks in intrusion detection systems. International Journal of Computer Networks and Applications 2017; 4 (2): 62-76. doi: 10.22247/ijcna/2017/48968

[2] Baykara M and Das R. SoftSwitch: a centralized honeypot-based security approach using software-defined switching for secure management of VLAN networks. Turkish Journal of Electrical Engineering & Computer Sciences 2019; 27 (5): 3309-3325. doi: 10.3906/elk-1812-86

[3] Zuech R, Khoshgoftaar TM, Wald R. Intrusion detection and big heterogeneous data: a survey. Journal of Big Data 2015; 2 (1): 1-41. doi: 10.1186/s40537-015-0013-4

[4] White T. Hadoop: The Definitive Guide. Massachusetts, USA: O'Reilly Media Incorporated Company, 2012.

[5] Salloum S, Dautov R, Chen X, Peng PX, Huang JZ. Big data analytics on Apache Spark. International Journal of Data Science and Analytics 2016; 1 (3): 145-64. doi: 10.1007/s41060-016-0027-9

[6] Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J et al. Cudnn: Efficient primitives for deep learning. Arxiv preprint 2014; arxiv: 1410.0759.

[7] Lunt TF. A survey of intrusion detection techniques. Computers & Security 1993; 12 (4): 405-418.

[8] Tan J, Pan X, Kavulya S, Gandhi R, Narasimhan P. SALSA: analyzing logs as state machines. Carnegie Mellon University Research Centers and Institutes, Pittsburgh, USA, 2008.

[9] Xu W, Huang L, Fox A, Patterson D, Jordan MI. Detecting large-scale system problems by mining console logs. In: Proceedings of the 27 th International Conference on Machine Learning; Haifa, Israel; 2010. pp. 117-132.

[10] Lee Y, Lee Y. Toward scalable internet traffic measurement and analysis with hadoop. ACM SIGCOMM Computer Communication Review 2012; 43 (1): 5-13. doi: 10.1145/2427036.14 2427038

[11] Suthaharan S. Big data classification: problems and challenges in network intrusion prediction with machine learning. ACM SIGMETRICS Performance Evaluation Review 2014; 41 (4): 70-73. doi: 10.1145/2627534.2627557

[12] Desai AS, Gaikwad DP. Real time hybrid intrusion detection system using signature matching algorithm and fuzzy-GA. In: IEEE international conference on advances in electronics communication and computer technology (ICAECCT); Pune, India; 2016. pp. 291-294.

[13] Marir N, Wang H, Feng G, Li B, Jia M. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. IEEE Access 2018; 6: 59657-59671. doi: 10.1109/ACCESS.2018.2875045

[14] Karatas G, Demir O, Sahingoz OK. Deep learning in intrusion detection systems. In: International Congress on Big Data Deep Learning and Fighting Cyber Terrorism (IBIGDELFT); Ankara, Turkey; 2018. pp. 113-116.

[15] Lovick C. The BSD syslog Protocol. Network Working Group, Grossrinderfeld, DE, 2001.

[16] Stallings W. SNMP and SNMPv2: the infrastructure for network management. IEEE Communications Magazine 1998; 36 (3): 37-43. doi: 10.1109/35.663326

[17] Estan C, Keys K, Moore D, Varghese G. Building a better NetFlow. ACM SIGCOMM Computer Communication Review 2004; 34 (4): 245-256. doi: 10.1145/1030194.1015495

[18] Suriadi S, Andrews R, ter Hofstede AH, Wynn MT. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information Systems 2017; 64: 132-150. doi: 10.1016/j.is.2016.07.011

[19] Rabkin A, Katz R. Chukwa: a system for reliable large-scale log collection. In: LISA'10 24th Large Installation System Administration Conference; San Diego, CA, USA; 2010. pp. 163-177.

[20] Groeneveld RA, Meeden G. Measuring skewness and kurtosis. Journal of the Royal Statistical Society: Series D (The Statistician) 1984; 33 (4): 391-399.

[21] Mardia KV. Measures of multivariate skewness and kurtosis with applications. Biometrika 1970; 57 (3): 519-530. doi: 10.1093/biomet/57.3.519

[22] Li C, Ren J, Huang H, Wang B, Zhu Y, Hu H. PCA and deep learning based myoelectric grasping control of a prosthetic hand. Biomedical Engineering 2018; 17 (1): 1-8. doi: 10.1186/s12938-018-0539-8

[23] Gulli A, Pal S. Deep learning with Keras. Birmingham, UK: Packt Publishing Ltd, 2017.

[24] Gardner MW, Dorling SR. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment 1998; 32 (14-15): 2627-2636. doi: 10.1016/S1352-2310(97)00447-0

[25] Eckle K, Schmidt-Hieber J. A comparison of deep networks with ReLU activation function and linear spline-type methods. Neural Networks 2019; 110: 232-242. doi: 10.1016/j.neunet.2018.11.005

[26] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: IEEE symposium on computational intelligence for security and defense applications; Ottawa, ON, Canada; 2009. pp. 1-6.

[27] Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018); Fredericton, Canada; 2018. pp. 108-116.

[28] Sharafaldin I, Gharib A, Lashkari AH, Ghorbani AA. Towards a reliable intrusion detection benchmark dataset. Software Networking 2018; 2018 (1): 177-200. doi:10.13052/jsn2445-9739.2017.009

[29] Noble WS. What is a support vector machine? Nature Biotechnology 2006; 24 (12): 1565-1567. doi: 10.1038/nbt1206-1565

[30] Le Roux N, Bengio Y. Representational power of restricted Boltzmann machines and deep belief networks. Neural Computation 2008; 20 (6): 1631-1649. doi: 10.1162/neco.2008.04-07-510