A Deep Learning Approach for Intrusion Detection

Roua Dhahbi[1], Farah Jemili[2]

[1]rouadhahbi@gmail.com [2]jmili_farah@yahoo.fr

[1]Higher Institute of Computer Science and Telecom (ISITCOM), University of Sousse, Hammam Sousse 4011, Tunisia
[2]MARS Research Lab LR17ES05, Higher Institute of Computer Science and Telecom (ISITCOM), University of Sousse, Hammam Sousse 4011, Tunisia

Abstract

Intrusion detection has been widely studied in both industry and academia, but cybersecurity analysts always want more accuracy and global threat analysis to secure their systems in cyberspace. Big data represent the great challenge of intrusion detection systems, making it hard to monitor and analyze this large volume of data using traditional techniques. Recently, deep learning has been emerged as a new approach which enables the use of Big Data with a low training time and high accuracy rate. In this paper, we propose an approach of an IDS based on cloud computing and the integration of big data and deep learning techniques to detect different attacks as early as possible. To demonstrate the efficacy of this system, we implement the proposed system within Microsoft Azure Cloud, as it provides both processing power and storage capabilities, using a convolutional neural network (CNN–IDS) with the distributed computing environment Apache Spark, integrated with Keras Deep Learning Library. We study the performance of the model in two categories of classification (binary and multiclass) using CSE-CIC-IDS2018 dataset. Our system showed a great performance due to the integration of deep learning technique and Apache Spark engine.

Keywords

Intrusion Detection System; Deep Learning; Convolutional Neural Network; Apache Spark; Microsoft Azure Cloud.

I. INTRODUCTION

In recent years, the world has seen a significant evolution in the different areas of connected technologies such as smart grids, the Internet of vehicles, long-term evolution, and 5G communication. By 2023, it is expected that the number of IP-connected devices will be three times larger than the global population, and the total number of DDoS attacks will double from 7.9 million in 2018 to 15.4 million by 2023 as reported by Cisco [1].

As of 2020, the amount of data generated each day is exceeding petabytes and this includes the traces that internet users make when they access a website, mobile application or a network. This growth gave more space to hackers to launch their malicious attacks and use development techniques and tools for intrusion. Intrusion detection systems (IDSs) are one of the most important systems used in cyber security. Intrusion detection systems are the hardware or software that monitors and analyzes data flowing through computers and networks to detect security breaches that threaten confidentiality, integrity and availability of a system's resources [2]. According to IBM [3], the costs of data breaches have increased from $3.86 million to $4.24 million in 2021. This is a loss that no business would be able to sustain. That's why, as Forbes reports [4], 83% of enterprise workloads will move to the cloud by 2020, making it necessary to develop new and efficient IDS.

Deep learning for intrusion detection is one of the hot topics in recent academic research. With the improvement of computing power and the rapid growth of the volume of data, the development of deep learning have attracted people's attention again, so that the practicality and popularity of deep learning have greatly improved [5]. Deep learning is an advanced branch of machine learning that uses multi layer networks. The layers are connected by neurons, which represent the mathematical computation of the learning processes [6]. Dong and Wang conducted a literature and experimental comparison between using specific traditional NIDS techniques and deep learning methods [7]. The authors concluded that the deep learning-based methods provided improved detection accuracy. Compared to traditional machine learning algorithms, deep learning techniques have a faster processing speed when dealing with big data and can directly learn feature representations from the raw data, like images and texts, without requiring manual feature engineering. Deep learning models consist of diverse deep networks which are classified into three main categories [8]: 1) generative architectures, which apply unsupervised learning to learn automatically from an unlabelled dataset among them Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs), Deep Boltzmann Machines (DBMs) and Deep Auto Encoders (DA), 2) discriminative architectures, which apply supervised learning mainly to distinguish patterns for prediction tasks among them Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) and 3) hybrid architectures, which incorporate both generative and discriminative models such as Generative Adversarial Networks (GANs). In [9] the most used models in IDS are: auto-encoders (AE), Convolutional Neural Network (CNN), Restricted

Manuscript received October 5, 2023
Manuscript revised October 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.10.12

Boltzmann Machine (RBM) and Recurrent Neural Network (RNN). Ferrag et al. [8] analyzed seven deep learning approaches, including RNN, DNN, RBM, DBN, CNN, DBM, and DA using the CSE-CIC-IDS2018 and Bot-IoT datasets. The experimental results showed that in the deep discriminative models, CNN gets a higher accuracy and a better mean false alarm rate in the two datasets, when there are 100 hidden nodes and a learning rate is equal to 0.5. Kim et al. [10] showed that CNN model performs better than the RNN model when applied to CSE-CIC-IDS 2018 dataset. In addition, CNN is also implemented on the KDD-CUP99 dataset in [11]. The authors claim that convolutional neural network has a superior performance then RNN and DNN.

Based on the methods and shortcomings of existing works, we propose a new approach of intrusion detection implemented within a cloud infrastructure, based on the integration between big data management and computation and deep learning techniques using Apache Spark and the Keras deep learning library. A convolutional neural network (CNN) is used. We study the performance of the model in binary and multiclass classification using a recent dataset CSE-CIC-IDS2018.

This paper organized as follows: In Section 2, there is a review of the related research which deals with IDS. Moreover, Section 3 presents the proposed approach and its components. Section 4 provides the evaluation metrics and results of the tested system, followed by Section 5 with a summary and discussion. Finally, Section 6 shows conclusions and offers new possibilities for the development of future work.

II. LITERATURE REVIEW

A broad range of research has been conducted on intrusion detection to find new models for more effective and performant. In this section, we present related works and current methods used for intrusion detection.

Belouch et al. [12] tested the performance of intrusion detection by applying Apache Spark and four supervised machine learning algorithms on the complete UNSW-NB15 dataset. The results indicate that the Random Forest classification algorithm gave the better performance in terms of accuracy 97.49% and prediction time 0.08 seconds.

Dong et al. [13] proposed a real-time Intrusion Detection System using deep learning and Big data processing capabilities. Flume, Flink, and a deep learning algorithm called Auto-Encoder are employed. The proposed approach has shown high performances, about 94.32% of accuracy using the KDD 99 dataset.

In [14], Big Data and Deep Learning Techniques are combined to improve intrusion detection system performance. Deep Feed-Forward Neural Network (DNN) and two ensemble approaches, Random Forest (RF) and Gradient Boosting Tree (GBT), are used to classify the CICIS2017 and UNSW NB15 datasets, which are implemented using the distributed computing environment Apache Spark. The experimental results show a high accuracy with DNN for binary and multiclass classification on UNSW NB15 dataset. With the CICIDS2017 dataset, the GBT classifier achieved the best binary classification accuracy, while DNN had the greatest multiclass classification accuracy.

Hafsa et al. [15] proposed a cloud based distributed IDS that uses Apache Spark Structured Streaming to detect anomalies in real time. Their proposed system shows that Decision Tree classifier yields a 99.95 % accuracy using a daily updated dataset MAWILab and can also process more than 55,175 records in one second using only a two worker-nodes cluster. This research has been executed by using simulation in the Microsoft Azure platform.

Lin et al [16] used LSTM to build a model for attack detection and attack type classification. The model provides 96.2% accuracy on the CSE-CIC-IDS2018 dataset. The researchers used the SMOTE oversampling technique to handle class imbalance by generating synthetic samples for minority classes. They also used an attention mechanism (AM) to improve model performance.

Kim et al [10] compared the performance of CNN and RNN deep learning methods on the CSE-CIC-IDS2018 dataset. The experimental results show that the CNN model is more accurate than the RNN model and achieves a detection accuracy of 0.9677 on the SD-3 sub-dataset.

In [17], Haggag et al. proposed DLS-IDS, a deep learning based IDS. IDS was implemented on Apache Spark using three DL models, namely Multilayer perceptron (MLP), Recurrent Neural Network (RNN), and Long-Short Term Memory (LSTM). By performing necessary experiments on the NSL-KDD dataset, the IDS approach achieves a detection accuracy of 83.57%.

Pham et al [18] suggested a lightweight intrusion detection system that transformed network traffic into two-dimensional image data and then used a convolutional neural network (CNN) to detect intrusions. Using the CSE-CIC-IDS2018 dataset, the model achieved an accuracy value of 95% with a reasonable detection time.

In [19] a convolutional recurrent neural network (CRNN) is used to construct a DL-based hybrid ID framework that predicts and classifies malicious cyberattacks in the network using the CSE-CIC-DS2018 dataset. The simulation results prove that the proposed HCRNNIDS has high performance in detecting both local features and temperature features and gives an accuracy of 97.75%.

III. RESEARCH METHODOLOGY

Our approach involves several steps, from data ingestion to attack detection. Two major sections are discussed here. The first part tackles the extraction of data files from CSE-CIC-IDS2018 dataset, followed by the preparing and the cleaning of these files. The second section will handle binary and multiclass classification of data using a convolutional neural network. Figure 3 shows the overall process.

A. Data and Methods

1) Dataset Description: In order to test the efficiency of such mechanisms, reliable datasets are needed that contain both benign and several attacks, meet real world criteria, and publicly available [20]. In our work, we use a new real traffic data set CSE-CIC-IDS2018 [21] developed by the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC) on AmazonWeb Services (AWS) [8]. CSE-CIC-IDS2018 includes seven different attack scenarios such as Heartbleed, Brute-force, DoS, DDoS, Web attacks, Botnet, and infiltration that are briefly described below. This dataset was generated using CICFlowMeter-V3 [22] and contains 80 features of raw network data that were saved in 10 csv files. The details of the features can be found at the website of CIC-2018 [21].

Brute Force Attack: This is one of the most popular attacks that only cannot be used for password cracking, but also to find hidden pages and content in a web page. The dataset includes brute force attacks on two current network services, SSH and FTP, with traffic data labeled as "SSH-Bruteforce" and "FTP-Bruteforce" respectively.

Denial of Service (DoS) attack: DoS attack is a very common type of network attack where attackers seeks to make a machine or network resource unavailable temporarily by flooding the targeted machine or resource with overwhelming number of fake requests in an attempt to overload systems and prevent some or all legitimate requests from being executed. To produce DoS attack traffic, the authors of the datasets used widely available programs such as Hulk, GoldenEye, Slowloris, and Showhttptest [20].

Distributed Denial of Service (DDoS) attack: It is more sophisticated denial of service attack where attackers use multiple compromised systems (for example, a botnet) to flood the targeted system by generating overwhelming amount of network traffic. High Orbit Ion Canon (HOIC) and Low Orbit Ion Cannon (LOIC) were used to perform DDoS attack and generate the traffic captured in the dataset [20].

Heartbleed Attack: It is also categorized into a Dos attack [23]. It stems from a flaw in the OpenSSL cryptography library, which is a widely used Transport Layer Security (TLS) implementation. To elicit the victim's response, a corrupted heartbeat request with a short payload and large length field is often sent to the vulnerable party (typically a server).

Botnet attack: Botnet traffic was generated with the Ares tool, which is based on Python and can provide remote shell, file download, screenshot capture, and key logging [20].

Infiltration attack: In this scenario, we use the Metasploit as the most common security issues and vulnerability verifier. After victim download the infected file from Dropbox for windows machine or from infected USB flash memory for macintosh machine, now we can conduct different attacks on the victim’s network include full port scan, IP sweep and service enumerations using Nmap. The authors used Kali Linux as an attacker and the victims are Windows, Ubuntu and Macintosh systems in the Victim-Network.

2) Apache Spark: Apache Spark is a distributed engine, fast, flexible, and fault-tolerant for large-scale data processing. It was originated at the University of California Berkeley in the AMP Lab in 2009 [24]. It is an open-source cluster-computing framework for processing massive amounts of data. The main characteristic of Spark is the in-memory computation, which runs all programs up to 100 xs faster in memory, or 10 xs faster on disk, than Apache Hadoop. Apache Spark in Azure HDInsight is the Microsoft implementation of Apache Spark in the cloud [25]. HDInsight makes it simple to create and configure a Spark cluster in Azure. Spark clusters in HDInsight are compatible with both Azure Blob Storage and Azure Data Lake Storage (Gen1 or Gen2). Thus, you can use HDInsight Spark clusters to process your data stored in Azure. It is based on Scala, but offers APIs for Java, Python and R. Spark has a diversified ecosystem that includes the following components: Spark SQL and DataFrames, Spark Streaming for real-time stream processing, MLlib for machine learning, and GraphX for graph processing. All the functionalities provided by

Apache Spark are built on the top of Spark Core which is the foundation of parallel and distributed processing of enormous datasets as shown in Figure 1.

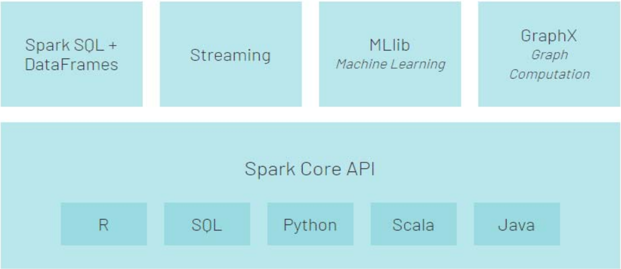


Fig. 1. Apache Spark Ecosystem [24]

3) Microsoft Azure: Microsoft Azure, formerly known as Windows Azure, is a cloud platform for building, deploying, and managing services and applications, anywhere with the help of a global network of managed data centers located in 60+ regions around the world [26]. Microsoft maintains a growing directory of Azure services, ordering from storage, analytics, and virtual computing to big data enterprise management solutions, machine learning tools, and much more. Microsoft has designed Azure HDInsight [27] based on Hortonworks Data Platform (HDP) which contributes hugely towards making HDInsight a cloud-based service for processing and analysis of large volumes of and historical data. HDInsight clusters store data managed by Azure separately in Azure Storage Blobs or Azure Data Lake instead of HDFS who are protected by the Azure Active Directory and persists even after the cluster is deleted. Moreover, users can use the most popular open-source frameworks such as Hadoop, Spark, Storm, Hive and other products on HDInsight.

4) CNN: Convolutional neural networks (CNNs) are a well-known deep learning model proposed for image classification that is suitable for the massive network environment. The CNN consists of an input layer, a convolution layer that extracts the features of the image, a pooling layer that reduces the computation and improves the detection efficiency of the model by reducing the dimension of the extracted features, a fully connected layer that determines which class the input image belongs to and an output layer. In this work we used Convolutional Neural Networks due to its high prediction and effectiveness. Furthermore, compared with other DL algorithms, the greatest advantage of CNN is that it shares the same convolutional kernels, which would reduce the number of parameters and calculation amount of training once greatly; it would also allow it to recognize attack types of traffic data more quickly.

B. Collect and Prepare

Collect data: This step consists of collecting the CSV files of our dataset and loading it into Microsoft Azure Blob Storage [28]. It is massively scalable object storage service for unstructured data, where the data can be exposed to the public or stored privately. The operations will be executed on Apache Spark cluster in Azure HDInsight which uses Azure Blob Storage as cluster storage.

Data Preparation: In this step, we proceed to create an HDInsight Spark Cluster in which we can perform data manipulation in a distributed manner. We will be utilizing the Python API (PySpark) and Jupyter Notebook with Apache Spark.

To provide more suitable data for the neural network classifier, the dataset is subjected to a group of preprocessing operations. These operations are summarized below:

- **Label Encoding:** The multi-class labels in the dataset are supplied with attack names, which are string values. It is therefore important to encode these values into numeric values, so that the classifier can learn the class number to which each tuple belongs.

- **Data normalization:** The numerical data in the dataset has a wide range of values, which makes it difficult for the classifier during training to compensate for these differences. The min-max normalization method is implemented to normalize the value of each feature in the range [0,1]. The transformation function is presented in (1) where xi represents the feature. Max and Min represent the maximum and minimum value of the feature.

$$\mathbf{Xi} = \frac{\mathbf{Xi - Min}}{\mathbf{Max - Min}} \quad (1)$$

- **Fill missing values:** This step consists of filling in the missing values, i.e. instead of deleting the records containing missing values; we have proposed to fill them with default values, the value "0" for the attributes of type Integer and the value "unknown" for the attributes of type String in order to complete the dataset.

- **Eliminating redundancies:** The alert database contains several redundant connections that have been captured multiple times. This problem causes an algorithm bias towards frequent registrations which prevents the algorithm from learning rare recordings, which are generally more harmful to networks. This problem is mitigated by deleting all repeated records in the Learning and Test Set by keeping only one copy of each record. This strategy renders the computation faster because it must process less data with reduce the size of dataset.

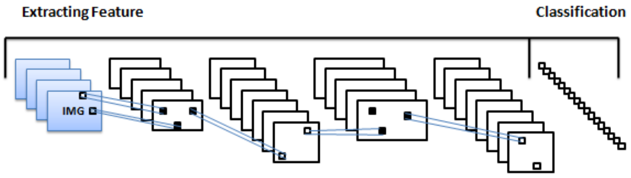
Converted into image data: CNN is the most commonly used deep learning algorithm for image classification. Therefore, to reduce the computational cost, we convert the CSE-CIC-2018 dataset into images. We convert each labeled data into 13x6 size of images because each data contains 78 features except the 'Label' feature and with the exception of 'Timestamp'. The 'Label' is used for image classification. The details of the features can be found at the website of CSE-CIC-2018 [21].

C. *CNN Intrusion Detection Model*

After preparing the data, we proceed by loading our CNN model. We deploy two convolutional layers and two maxpooling layers where each convolution layer and each pooling layer are set alternately to accurately and efficiently extract the intrusion characteristics. In addition, we use 'relu' as an activation function for each convolutional layer because of its faster training speed. Then, dropout is applied after each step of the max pooling in order to reduce overfitting. Finally, a fully connected layer is deployed behind the last max-pooling layer where the 2D image matrix is converted to a 1D feature vector as the input.

For the output layer, we use the sigmoid function for binary classification, while the SoftMax function is used for multiclass classification.

By organizing those layers along with modeling parameters such as a kernel size, number of kernels, and ratio of dropout we can find out the optimal CNN model. Figure 2 shows the detailed parameter of our CNN model.



	Convolution Layer1	Pooling Layer1	Convolution Layer2	Pooling Layer2	Fully-connected Layer
Num. of kernels	16	-	32	-	-
Size of kernel	3*3	-	3*3	-	-
Size of pooling	-	2*2	-	2*2	-
Dropout	-	0.6	-	0.6	0.6

Fig. 2. Our CNN Model and Parameters

After determining the optimal parameters of the network model, the performance of the model is evaluated by the classification results of the test dataset. The overall workflow of our proposed approach is illustrated in Figure 3. Our model has been registered in Azure Blob Storage for future use. Thus, it can be updated manually to train on new data and can be exported again to replace the old model.

In the next chapter, we report the experimental results of our distributed system in two scenarios (binary classification and multi-class classification) and compare it to traditional IDS.

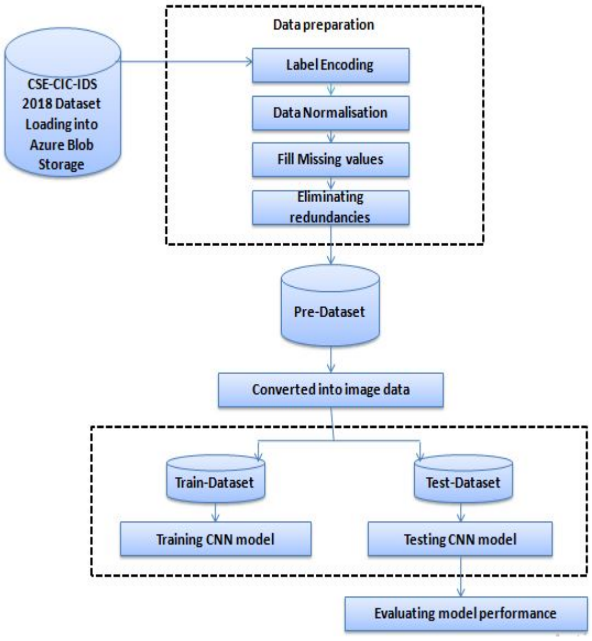


Fig. 3. Proposed Approach

IV. EXPERIMENTAL RESULTS

A. *Experiment Setup*

This section introduces the results of evaluations for the two scenarios, namely the binary and multi-class classifications. The experiments are conducted on Microsoft Azure HDInsight running on Linux virtual machines and Spark version 3.0.0 running on top of YARN, using Jupyter Notebook and Python API (Pyspark). Also, our approach is implemented using the distributed Keras library.

The characteristics of our spark cluster are illustrated in Table 1.

TABLE I
SETUP USED TO TEST OUR SOLUTION

	HEAD NODE	WORKER NODE
Number	2	2
Name	D3 V2 optimized	D4 V2 optimized
CPU	4 vCPUs	8 vCPUs
Memory (RAM)	14 GB	28 GB
Operating System	Linux (CentOS) x64bit	Linux (CentOS) x64bit
Storage	200 GB SSD	400 SSD

We split the dataset into two subsets, one used to train the model and the other to evaluate it. In our experiment, 70% for training and 30% for testing. We set the training parameters as shown in Table 2.

TABLE II
TRAINING PARAMETERS FOR OUR EXPERIMENTS

PARAMETERS	VALUE
Learning rate	0.1
Optimization algorithm	Adam
Size of batch	100
Number of epochs	10

B. *Performance Metrics*

The performance measures [29] used in order to verify the performances of our model are listed in table 3 given below.

TABLE III
EVALUATION METRICS

Measure	Description	Formula
Accuracy (Ac)	The ratio of correctly classified samples to total samples	Ac=(TP+TN)/TP+FN+ TN+FP
Precision (P)	The ratio of true positive samples to predicted positive samples	P=TP/TP+FP
Recall (R)	The ratio of true positive samples to total positive samples	R=TP/TP+TN
F-measure (F)	The harmonic average of the precision and the recall	F=2(P*R)/(P+R)
Detection rate (DR)	The ratio of the number of correctly detected attacks to the total number of attacks	DR=TP/TP+FN
False Positive Rate (FPR)	The ratio of the number of normal instances detected as attack to the total number of normal instances	FPR=FP/FP+TN

Where TP, FP, TN, FN represent True Positive, False Positive, True Negative and False Negative respectively.

C. *Binary Classification*

Table 4 shows the results of the binary classification. Our model obtained good results with 99.85% accuracy.

TABLE IV
MODEL EVALUATION MATRICS FOR BINARY CLASSIFICATION

Ac	P	R	F	DR	FPR
99.85%	99.76%	99.79%	99.8%	99.87%	0.03%

D. *Multiclass Classification*

For multiclass classification, the results show that our model has an accuracy of (99.98%) as shown in the following Table 5.

TABLE V
PERFORMANCE METRICS RELATIVE TO THE DIFFERENT ATTACK TYPE AND BENIGN

	P	R	F	DR	FPR	Ac
Benign	0.98	0.97	0.97	0.99	0.02	99.98%
Brute-force	0.97	0.96	0.96	0.98	0.028	
Web attack	0.99	0.98	0.98	0.99	0.019	
DoS attack	1	1	1	1	0.001	
DDoS attack	0.96	0.93	0.94	0.97	0.027	
Botnet	0.89	0.84	0.86	0.91	0.04	
Infilteration	0.88	0.86	0.87	0.89	0.046	

V. DISCUSSION

The results obtained indicate that our solution is realistic, since the intrusion detection rate is increased and the false positive rate is decreased. This means that our system can detect a large number of attacks in a relatively short time. When we compare the results in the two scenarios, we see an improvement in the accuracy and prediction time of the multi-class classification compared to the binary classification.

Table 6 summarizes the results obtained with the existing methods for the CSE-CIC-IDS2018 dataset. Since these datasets were created after the DARPA, KDD and several other data sets, some preliminary results are available. Our proposed model outperforms the state-of-the-art techniques in terms of accuracy and this is due to the execution of the deep learning approach.

In addition, the use of apache spark has significantly improved the training and prediction time of our model compared to traditional techniques. This enhancement gives intrusion detection systems the ability to make more effective decisions about whether to block or allow data to pass through a network. Furthermore, the integration between Apache Spark and the Keras deep learning library has increased the capabilities of deep learning algorithms to work more efficiently and quickly.

Although our proposed model outperforms the evaluated metrics, it is difficult to believe that the proposed approach totally outperformed the other approaches. But with the proposed solution, we assert that one can reach an exceptional amount of network protection and easily identify malicious threats.

TABLE VI
COMPARISON OF EXISTING APPROACHES WITH THE CSE-CIC-IDS2018 DATASET

References	Year	Methods	Accuracy
Lin et al. [16]	2019	LSTM	96.2%
Kim et al. [10]	2019	CNN	96.77%
Pham et al. [18]	2020	CNN	95%
Khan et al. [19]	2021	HCRNN	97.75%

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new architecture to improve the performance of intrusion detection systems by integrating big data technologies and deep learning techniques. CSE-CIC-IDS-2018 dataset was employed to evaluate the performance of the proposed approach against cyber-attacks. In our method we used the convolutional neural network (CNN) classifier to classify the attacks in binary and multiclass modes. In order to overcome the problems of storing large datasets and to accelerate the processing speed of data, this system is used with Microsoft Azure to showcase its performance within a distributed cloud infrastructure using Apache Spark with the Keras Deep Learning Library. The experimental results show high accuracy levels for binary and multiclass classification (99.85% and 99.98% respectively). For future work, we propose to use multiple clusters with varying node counts to achieve faster results and combine two or more datasets. Moreover, in order to obtain a more accurate and powerful system, we propose to process data in real time (real-time IDS).

References

[1] "Cisco Cybersecurity Reports," [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accessed 15 02 2021].

[2] R. Di Pietro and L. V. Mancini, Intrusion detection systems, vol. 38, Springer Science & Busines Media, 2008.

[3] "How much would a data breach cost your business?, " [Online]. Available: https://www.ibm.com/security/data-breach. [Accessed 12 04 2021].

[4] "83% Of Enterprise Workloads Will Be In The Cloud By 2020, " [Online].Available:https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/5ea5c4696261. [Accessed 15 04 2021].

[5] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436--444, 2015.

[6] Y. Bengio, I. Goodfellow and A. Courville, Deep learning, MIT Press, 2016.

[7] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), pp. 581—585, 2016.

[8] M. A. Ferrag, L. Maglaras, S. Moschoyiannis and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," Journal of Information Security and Applications, vol. 50, p. 102419, 2020.

[9] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang and R. X. Gao, "Deep learning and its applications to machine health monitoring: A survey," arXiv preprint arXiv:1612.07640, 2016.

[10] J. Kim, Y. Shin and E. Choi, "An intrusion detection model based on a convolutional neural network," Journal of Multimedia Information System, vol. 6, pp. 165--172, 2019.

[11] Y. Xiao, C. Xing, T. Zhang and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," IEEE Access, vol. 7, pp. 42210--42219, 2019.

[12] M. Belouch, S. El Hadaj and M. Idhammad, "Performance evaluation of intrusion detection based on machine learning using apache spark," Procedia Computer Science, vol. 127, pp. 1--6, 2018.

[13] Y. Dong, R. Wang and J. He, "Real-Time Network Intrusion Detection System Based on Deep Learning," in 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), IEEE, pp. 1—4, 2019.

[14] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," in Proceedings of the 2019 ACM Southeast Conference, pp. 86—93, 2019.

[15] M. Hafsa and F. Jemili, "Comparative study between big data analysis techniques in intrusion detection," Big Data and Cognitive Computing, vol. 3, p. 1, 2019,

[16] P. Lin, K. Ye and C. Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in International conference on cloud computing, Springer, pp. 161—176, 2019.

[17] M. Haggag, M. M Tantawy and M. MS El-Soudani, "Implementing a deep learning model for intrusion detection on apache spark platform," IEEE Access, vol. 8, pp. 163660—163672, 2020.

[18] V. Pham, E. Seo, T. M. Chung, "Lightweight Convolutional Neural Network Based Intrusion Detection System," J. Commun., vol.15, pp. 808--817, 2020.

[19] M. A. Khan, "HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System, " Processes, Multidisciplinary Digital Publishing Institute, vol. 9, pp. 834, 2021.

[20] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in ICISSP, pp. 108—116, 2018.

[21] "CSE-CIC-IDS2018 on AWS," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html. [Accessed 15 03 2021].

[22] "CICFlowMeter," [Online]. Available: https://www.unb.ca/cic/research/applications.html#CICFlowMeter. [Accessed 16 03 2021].

[23] G. Karatas, O. Demir and O. K. Sahingoz, "Deep Learning in Intrusion Detection Systems," in 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), IEEE, pp. 113—116, 2018.

[24] "Apache Spark," [Online]. Available: https://databricks.com/spark/about. [Accessed 07 03 2021].

[25] "What is Apache Spark in Azure HDInsight," [Online]. Available:https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-overview.[Accessed 20 03 2021].

[26] "Azure geographies," [Online]. Available: https://azure.microsoft.com/en-us/global-infrastructure/geographies/. [Accessed 13 03 2021].

[27] "What is Azure HDInsight?," [Online]. Available: https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-overview. [Accessed 14 03 2021].

[28] "Blob storage," [Online]. Available: https://azure.microsoft.com/en-us/services/storage/blobs/. [Accessed 27 02 2021].

[29] "Evaluation Metrics - RDD-based API," [Online].Available: https://spark.apache.org/docs/2.1.0/mllib-evaluation-metrics.html. [Accessed 18/04/2021].