Intrusion Detection Using Deep Learning

Md Ariful Haque
Modeling Simulation and Visualizationg Engineering
Old Dominion University
Norfolk, VA, USA
mhaqu001@odu.edu

Jiang Li
Electrical and Computer Engineering
Old Dominion University
Norfolk, VA, USA
jli@odu.edu

Abstract—**With the enormous growth of computer networks usage and the huge increase in the number of applications running on top of it, network security is becoming increasingly more important. During the last decade, anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based intrusion detection systems (IDSs) in detecting novel attacks, and KDD CUP 99 is the mostly widely used data set for the evaluation of these systems. Having conducted a statistical analysis on this data set, researchers found some issues which highly affects the performance of evaluated systems, and results in a very poor evaluation of anomaly detection approaches. To solve these issues, researchers have proposed a new data set, NSL-KDD, which consists of selected records of the complete KDD data set and does not suffer from those shortcomings. In recent times, data mining and machine learning have been subjected to extensive research in intrusion detection with emphasis on improving the accuracy of detection classifier. In this work, we have considered three different classifiers on these two data sets: Support Vector Machine (SVM), Convolution Neural Network (CNN), and fully connected Deep Neural Network (DNN). We briefly describe the data preprocessing techniques and model definition and implementation. Then we present our results and provide performance evaluation. Our results show that, DNN provides the best result in terms of accuracy and CPU computation time.**

Keywords—Intrusion Detection System, CNN, DNN, SVM

I. Introduction

An intrusion detection system (IDS) is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered [1]. While anomaly detection and reporting are the primary functions, some intrusion detection systems can take actions when malicious activity or anomalous traffic is detected, including blocking traffic sent from suspicious IP addresses. Although intrusion detection systems monitor networks for potentially malicious activity, they are also prone to false alarms (false positives). Consequently, organizations need to fine-tune their IDS products when they first install them. That means properly configuring their intrusion detection systems to recognize what normal traffic on their network looks like compared to potentially malicious activity. An intrusion prevention system (IPS), on the other hand, not only have the capability to monitors network packets for potentially damaging network traffic, but also can respond to such traffic by rejecting the potentially malicious packets where an intrusion detection system responds to potentially malicious traffic by logging the traffic and issuing warning notifications, intrusion prevention systems.

Intrusion detection systems (IDSs) identify activities that violate the security policy of a computer system or network. IDS is a necessary complement to preventive security mechanisms such as firewalls because IDS detect attacks that exploit system design flaws or bugs and IDS provide forensic evidence to inform system administrators reactions to cyberattacks [2]. Intrusion detection assumes that intrusive activities are noticeably different from normal system activities and thus detectable. Intrusion detection is not introduced to replace prevention-based techniques such as authentication and access control; instead, it is intended to complement existing security measures and detect actions that bypass the security monitoring and control component of the system. Intrusion detection is therefore considered as a second line of defense for computer and network systems. Generally, an intrusion would cause loss of integrity, confidentiality, denial of resources, or unauthorized use of resources. Some specific examples of intrusions that concern system administrators include [3]:

- Unauthorized modifications of system files to facilitate illegal access to either system or user information.
- Unauthorized access or modification of user files or information
- Unauthorized modifications of tables or other system information in network components (e.g. modifications of router tables in an internet to deny use of the network).
- Unauthorized use of computing resources (perhaps through the creation of unauthorized accounts or perhaps through the unauthorized use of existing accounts).

Some of the important features an intrusion detection system should possess include [4]:

- Be fault tolerant and run continually with minimal human supervision. The IDS must be able to recover from system crashes, either accidental or caused by malicious activity Possess the ability to resist subversion so that an attacker cannot disable or modify the IDS easily.
- Furthermore, the IDS must be able to detect any modifications forced on the IDS by an attacker.
- Impose minimal overhead on the system to avoid interfering with the normal operation of the system.

This report is based on the class project of the course Machine Learning I (MSIM 607).

- Be configurable to accurately implement the security policies of the systems that are being monitored. The IDS must be adaptable to changes in system and user behavior over time.
- Be general enough to detect different types of attacks and must not recognize any legitimate activity as an attack (false positives). At the same time, the IDS must not fail to recognize any real attacks (false negatives).

Intrusion detection systems come in different forms and detect suspicious activities using different methods which including the following [1]:

- Network Intrusion Detection System (NIDS): A NIDS is deployed at a strategic point or points within the network, where it can monitor inbound and outbound traffic to and from all the devices on the network.
- Host Intrusion Detection System (HIDS): HIDS run on all computers or devices in the network with direct access to both the internet and the enterprise internal network. HIDS have an advantage over NIDS in that they may be able to detect anomalous network packets that originate from inside the organization or malicious traffic that a NIDS has failed to detect. HIDS may also be able to identify malicious traffic that originates from the host itself, as when the host has been infected with malware and is attempting to spread to other systems.
- Signature-based intrusion detection systems monitor all the packets traversing the network and compares them against a database of signatures or attributes of known malicious threats, much like antivirus software.
- Anomaly-based intrusion detection systems monitor network traffic and compare it against an established baseline, to determine what is considered normal for the network with respect to bandwidth, protocols, ports and other devices. This type of IDS alerts administrators to potentially malicious activity.

The rest of the paper are organized as follows. Section II provides a brief literature review on the use of machine learning and KDD CUP data in intrusion detection systems. Section III provides a brief methodology of data collection and section IV provides a description on the classifiers that are used in this project. The experiments and results are discussed in section V. Finally, Section VI concludes this paper.

II. Literature Review

The construction of traditional intrusion detection systems (IDSs) that use manually created rules based upon expert knowledge is knowledge-intensive and is not suitable in the context of this big data problem. Their strengths depend largely on the ability of the security personnel that develops them. The former can only detect known attack types and the latter is prone to generation of false positive alarms. This leads to the use of an intelligence technique known as data mining/machine learning technique as an alternative to expensive and strenuous human input. These techniques automatically learn from data or extract useful pattern from data as a reference for normal/attack traffic behavior profile from existing data for subsequent classification of network traffic.

In [5], the authors propose a deep learning based approach for developing such an efficient and flexible NIDS (Network Intrusion Detection System) where they use Self-taught Learning (STL), a deep learning based technique, on NSL-KDD - a benchmark dataset for network intrusion. In another work [6], the authors use support vector machines (SVM) for the classification. The verification regarding the effectiveness of the proposed system is done by conducting some experiments using NSL-KDD Cup'99 dataset which is improved version of KDD Cup'99 data set. In [7], the authors provide a detailed analysis on NSL-KDD dataset using various machine learning techniques. The authors [8] present the analysis of KDD data set with respect to four classes which are Basic, Content, Traffic and Host in which all data attributes can be categorized. The analysis is done with respect to two prominent evaluation metrics, Detection Rate (DR) and False Alarm Rate (FAR). There are several other works [9-14] which focus on the detailed analysis of the KDD Cup and NSL-KDD datasets, statistical analysis of the features of those datasets, and how the datasets are being used in IDS using various machine learning techniques.

Our contribution in this paper is summarized as follows:

- A concise description of the KDD and NSL-KDD Dataset
- Different classification techniques (SVM, CNN, and DNN) used for the IDS
- Preprocessing of the labeled datasets and the design of the classifiers along with the model specifications
- Performance evaluation of the above-mentioned classifiers for both the datasets.

III. Methodology

In this section, we illustrate different attack types and descriptions in the KDD Cup 99 and KDD-NSL dataset along with the preprocessing techniques. The next section contains the description of the machine learning techniques used to classify the different attack types.

A. KDD Cup 99 Dataset

The KDD Cup 1999 dataset [15] is a collection of nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN by MIT Lincoln Labs. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

The training data is made up of 22 different attacks out of the 39 presents in the test data. The known attack types are those present in the training dataset while the novel attacks are the additional attacks in the test datasets not available in the training data sets. The attacks fall into four main categories:

- **DOS:** denial-of-service, e.g. syn flood
- **R2L:** unauthorized access from a remote machine, e.g. guessing password
- **U2R:** unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow'" attacks
- **Probing:** surveillance and other probing, e.g., port scanning.

The KDD Cup training dataset consisted of 494,021 records among which 97,278 (19.69%) are normal, 391,458 (79.239%) DoS, 4,107 (0.83%) Probe, 1126 (0.23%) R2L, and 52 (0.0105%) U2R connections. Each connection consists 41 attributes describing different features of the connection and a label assigned to each either as an attack type or as normal. Table I shows the class labels and the number of samples that appears in 10% KDD training dataset.

B. NSL-KDD Dataset

NSL-KDD [16] is a data set suggested to solve some of the inherent problems of the KDD'99 data set which are mentioned in [14]. Although, this new version of the KDD data set may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs, it is still considered to be applicable as an effective because of the lack of public data sets for network-based IDSs, it is still considered to be applicable as an

TABLE I: KDD CUP 1999 DATASET ATTACK TYPES

Category	**Sub-category**	**Numbers (/494021)**	**Percentage**
Normal	-	97278	19.691 %
DoS	Back, land, Neptune, pod, smurf, teardrop	391458	79.239 %
Probe	Ipsweep, nmap, portsweep, satan	4107	0.83 %
U2R	buffer_overflow, loadmodule, perl, rootkit	52	0.01052 %
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster	1126	0.2279 %

TABLE II: NSL-KDD DATASET (KDDTRAIN+.TXT) ATTACK TYPES

Category	**Sub-category**	**Numbers (/125972)**	**Percentage**
Normal	-	67342	53.4579%
DoS	Back, land, Neptune, pod, smurf, teardrop	45927	36.458 %
Probe	Ipsweep, nmap, portsweep, satan	11656	9.25 %
U2R	buffer_overflow, loadmodule, perl, rootkit	52	0.041 %
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster	995	0.7898 %

effective benchmark data set to help researchers compare different intrusion detection methods. Furthermore, the number of records in the NSL-KDD train and test sets are reasonable. This advantage makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research work will be consistent and comparable.

The NSL-KDD data set has the following advantages [17] over the original KDD data set:

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There are no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

In each record there are 41 attributes unfolding different features of the flow and a label is assigned to each sample either as an attack type or as normal. The details of the attributes namely the attribute name, their description and sample data are given in Table II. The features the NSL-KDD dataset are of different data types. The datatypes and feature numbers are given in the following table II. Apart from normal data, records for 39 different attack types exist in NSL-KDD dataset. All these attack types were grouped into four attack classes. The summary of the attack classes and their attack types are given in table II and a detailed information is available in KDD Train+ (NSL-KDD) training dataset.

C. Data Preprocessing

As the data are labeled data, we need to perform the numerical conversion and normalization on the datasets before applying any classification techniques. This section describes the preprocessing steps.

Preprocessing: The neural network classification uses only numerical values for training and testing. Hence a preprocessing stage is needed to convert the non-numerical values to numerical values. Two main tasks in pre-processing are [9]:

- Converting the non-numerical features in the dataset to numerical values. The features 2,3 and 4 namely the protocol type, service and flag were nonnumerical. These features in the train and test data set were converted to numerical types by assigning specific values to each variable (e.g. TCP = 1, UDP = 2 and ICMP = 3).
- Convert the attack types at the end of the dataset into its numeric categories. 1 is assigned to normal data. 2, 3, 4 and 5 are assigned to DoS, Probe, R2L and U2R attack types. We considered 38 features for KDD CUP99 and 39 features.

Normalization: Since the features of both KDD and NSL-KDD dataset have either discrete or continuous values, the ranges of the features value were different, and this made them incomparable. As a result, the features are normalized by subtracting the mean from each feature and dividing by standard deviation.

D. Dataset Used For This Project

I have considered the following KDD and NSL-KDD datasets for this project results illustration:

- Train: kddcup.data_10_percent_corrected (KDD)
- Test: corrected_test.txt (KDD)
- Train: KDDTrain+ (NSL-KDD))
- Test: KDDTest+ (NSL-KDD))

IV. CLASSIFICATION MODELS

Under the tasks of the project, we have modeled a multi-class support vector machine (SVM) model, a convolutional neural network (CNN) model, and a deep learning (DNN) model. This section briefly describes the above three models and their results and perform a comparative analysis.

A. Multi-class SVM Model

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

SVMs are inherently two-class classifiers. The most common technique to do multiclass classification with SVMs in practice has been to build by one-versus-rest classifiers (commonly referred to as "one-versus-all" (OVA) classification), and to choose the class which classifies the test datum with greatest margin. Another strategy is to build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. In this project, we have used the LIBLENEAR library [18] for MATLAB which is designed to use for solving large scale classification problems. It supports L2-regularized logistic regression (LR), L2-loss and L1-loss linear support vector machines (SVMs). The results of SVM classification for the KDD Cup and NSL-KDD are presented in the Result section.

B. CNN Model

Convolutional Neural Networks (CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNNs have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars. There are four main operations in the CNN:

Convolution: The primary purpose of convolution in case of a CNN is to extract features from the input image or dataset. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. The size of the Feature Map (Convolved Feature) is controlled by three parameters: depth, stride, and zero-padding.

Non-Linearity (ReLU): An additional operation called ReLU are used after every convolution operation. ReLU stands for Rectified Linear Unit and is a non-linear operation. The purpose of ReLU is to introduce non-linearity in our CNN, since most of the real-world data we would want our CNN to learn would be non-linear.

Pooling or Sub Sampling: Spatial Pooling (also called subsampling or down sampling) reduces the dimensionality of each feature map but retains the most important information.

Fig. 1: CNN Model Summary (KDD Cup Dataset)

```
_________________________________________________________________
Layer (type)              Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)         (None, 1, 38, 64)         256
_________________________________________________________________
conv2d_2 (Conv2D)         (None, 1, 38, 64)         12352
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 19, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)         (None, 1, 19, 128)        24704
_____________________________________________________________
conv2d_4 (Conv2D)         (None, 1, 19, 128)        49280
_________________________________________________________________
conv2d_5 (Conv2D)         (None, 1, 19, 128)        49280
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 9, 128)      0
_________________________________________________________________
conv2d_6 (Conv2D)         (None, 1, 9, 256)         98560
_________________________________________________________________
conv2d_7 (Conv2D)         (None, 1, 9, 256)         196864
_________________________________________________________________
conv2d_8 (Conv2D)         (None, 1, 9, 256)         196864
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 4, 256)      0
_________________________________________________________________
flatten_1 (Flatten)       (None, 1024)              0
_________________________________________________________________
dense_27 (Dense)          (None, 100)               102500
_________________________________________________________________
dense_28 (Dense)          (None, 20)                2020
_________________________________________________________________
dense_29 (Dense)          (None, 5)                 105
=================================================================
Total params: 732,785
Trainable params: 732,785
Non-trainable params: 0
_______________________________________________________________
```

Spatial Pooling can be of different types: Max, Average, Sum etc. In case of Max Pooling, we define a spatial neighborhood (e.g., a 2×2 window) and take the largest element from the rectified feature map within that window. In practice, Max Pooling has been shown to work better.

Classification (Fully Connected Layer): The fully connected layer (dense layer) is a traditional Multi-Layer Perceptron that uses a Softmax activation function in the output layer. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer.

For the project, I have used the python Keras library to build our CNN model for both the datasets. Fig. 1 shows the model summary of the CNN model.

C. *DNN Model*

Deep Neural Network (DNN) have recently been achieving state-of-the-art performance on a variety of applications. DNNs learn in hierarchical layers of representation from inputs in order to perform necessary classification task. Recently, these deep learning architectures have demonstrated impressive and sometimes human competitive results in many applications. A DNN consists of convolutional, max-pooling and fully connected layers. All adjustable parameters are jointly optimized through minimization of the misclassification error over the training set. A DNN is a feed-forward, artificial neural network that has more than one layer of hidden units between inputs and its outputs. These multiple hidden layers can be useful for solving classification problems with complex data. Each layer can learn features at a different level of abstraction. Deep neural networks are often much harder to train than shallow neural networks.

For the project, I have used the python Keras library to build our DNN model for both the datasets. Fig. 2 shows the model summary of the CNN model.

Fig. 2: DNN Model Summary (NSL-KDD Dataset)

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_34 (Dense)          (None, 120)             4800
_________________________________________________________________
dropout_22 (Dropout)      (None, 120)             0
_________________________________________________________________
dense_35 (Dense)          (None, 80)              9680
_________________________________________________________________
dropout_23 (Dropout)      (None, 80)              0
_________________________________________________________________
dense_36 (Dense)          (None, 60)              4860
_________________________________________________________________
dropout_24 (Dropout)      (None, 60)              0
_________________________________________________________________
dense_37 (Dense)          (None, 5)               305
=================================================================
Total params: 19,645
Trainable params: 19,645
Non-trainable params: 0
_________________________________________________________________
```

TABLE III: SVM ACCURACY

Dataset	Training Accuracy	Testing Accuracy
KDD Cup (10%)	98.3022 %	94.4911 %
NSL-KDD	93.4923 %	69.8101 %

TABLE IV: CNN ACCURACY

Dataset	Training Accuracy	Testing Accuracy
KDD Cup (10%)	98.7692 %	93.6716 %
NSL-KDD	98.1160 %	82.5876 %

TABLE V: DNN ACCURACY

Dataset	Training Accuracy	Testing Accuracy
KDD Cup (10%)	98.6997 %	96.7806 %
NSL-KDD	99.3102 %	82.1616 %

The dropout layer is used to perform regularization which prevents overfitting. Also, stochastic gradient decent (SGD) is used to discriminative learning of linear classifiers under convex loss functions.

V. RESULTS AND COMPARISON

This section presents the training and testing accuracies for the SVM, CNN, and DNN classifiers that we have implemented.

A. *SVM Accuracy*

Table III shows the performance comparison of two datasets for SVM classifier. The accuracy of KDD cup is higher than NSL-KDD due to its redundant records. Therefore NSL-KDD datasets were introduced to ensure a better IDS system. In [14], the authors also get accuracies for NSL-KDD (KDDTest+.txt) around 69.52% which is very close to our results.

B. *CNN Accuracy*

Table IV shows the performance comparison of two datasets for CNN classifier. The accuracy results for KDD Cup is quite similar whereas the training and testing accuracy for NSL-KDD has increased to 98.1160% and 82.5876% respectively. According to [9], the training and testing accuracies for CNN using KDD Cup 99 dataset are over 98%. But they considered only 36 randomly selected features and 35000 randomly selected instances, whereas we used 38 features out of 41 and all the unique instances (around 143000 instances). That is why our test accuracy for KDD CUP was little bit lower (93.6716 %). We have considered three types of batch size (64, 128, and 256) and (3x3) size kernels. The number of epochs that we use in the simulation is 5.

C. DNN Accuracy

Table V shows the performance comparison of two datasets for DNN classifier. The results are almost similar to the CNN classifiers where the train accuracy for NSL KDD has increased to 99.3102 %. We have considered 3 hidden layers with 120, 80 and 60 hidden units respectively. According to [9],they got just over 98% training accuracy for NSL-KDD. We have considered 50 epochs for DNN.

D. CPU Computation Time Comparison

The SVM models take around 275 seconds to execute. The CNN models also take around 277 seconds per epoch. But the DNN models take only around 6 seconds per epoch. This means the DNN is much faster than both the SVM and CNN models. Thus, the DNN models are much less expensive than the CNN and SVM models in terms of CPU computation time and the accuracy of DNN is the highest among the three models.

VI. Conclusion

In this work, we summarize the implementation technique of three different classifiers (SVM, CNN, and DNN) for the IDS using KDD Cup and NSL-KDD dataset. Though the results presented here are good and comparable with results of other published articles, the classifiers are predicting majority class correctly. This is because both datasets are highly imbalanced, i.e., U2R class only stands for 0.04 % of NSL-KDD dataset and 0.01 % for KDD Cup. To overcome the problem of biased classification we can do any of the following:

- We can use either over or under sampling
- Assign greater weights to the minority class.

Overall, the project gives us a good experience to implement the machine learning knowledges in a practical problem.

Reference

[1] M. Rouse, "Unified threat management devices: Understanding UTM and its vendors," *SearchSecurity*, 2018.

[2] Pan, Shengyi, Thomas Morris, and Uttam Adhikari. "Developing a hybrid intrusion detection system using data mining for power systems." *IEEE Transactions on Smart Grid* 6, no. 6 (2015): 3104-3113.

[3] Alazab, Mamoun, Sitalakshmi Venkatraman, Paul Watters, and Moutaz Alazab. "Zero-day malware detection based on supervised learning algorithms of API call signatures." In *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121*, pp. 171-182. Australian Computer Society, Inc., 2011.

[4] A. L. Buczak, and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2016.

[5] Javaid, Ahmad, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. "A deep learning approach for network intrusion detection system." In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21-26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[6] Y. B. Bhavsar, and K. C. Waghmare, "Intrusion detection system using data mining technique: Support vector machine," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 581-586, 2013.

[7] S. Revathi, and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 12, pp. 1848-1853, 2013.

[8] P. Aggarwal, and S. K. Sharma, "Analysis of KDD dataset attributes-class wise for intrusion detection," *Procedia Computer Science*, vol. 57, pp. 842-851, 2015.

[9] L. Dhanabal, and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446-452, 2015.

[10] Kayacik, H. Günes, A. Nur Zincir-Heywood, and Malcolm I. Heywood. "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets." In *Proceedings of the third annual conference on privacy, security and trust*. 2005.

[11] Olusola, Adetunmbi A., Adeola S. Oladele, and Daramola O. Abosede. "Analysis of KDD'99 intrusion detection dataset for selection of relevance features." In *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, pp. 20-22. WCECS, 2010.

[12] Sahu, Santosh Kumar, Sauravranjan Sarangi, and Sanjaya Kumar Jena. "A detail analysis on intrusion detection datasets." In *2014 IEEE International Advance Computing Conference (IACC)*, pp. 1348-1353. IEEE, 2014.

[13] M. K. Siddiqui, and S. Naahid, "Analysis of KDD CUP 99 dataset using clustering based data mining," *International Journal of Database Theory and Application*, vol. 6, no. 5, pp. 23-34, 2013.

[14] Tavallaee, Mahbod, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. "A detailed analysis of the KDD CUP 99 data set." In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1-6. IEEE, 2009.

[15] "KDD Cup 1999 Data.", http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[16] "NSL-KDD Dataset.", https://github.com/defcom17/NSL_KDD

[17] UNB, "NSL-KDD dataset.", https://www.unb.ca/cic/datasets/nsl.html

[18] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871-1874, 2008.