A Comparative Study of Machine Learning and Deep Learning in Network Anomaly-Based Intrusion Detection Systems

Mohab Sameh Abdel-Wahab
Faculty of Computer Science,
October University for Modern
Sciences and Arts (MSA), Egypt
mohab.sameh@msa.edu.eg

Ahmed M. Neil
Faculty of Computer Science,
October University for Modern
Sciences and Arts (MSA), Egypt
CIO Cyber Security Threat
Hunter and Engineer
IBM Bratislava, Slovakia
Ahmed.neil@ibm.com

Ayman Atia
Faculty of Computer Science,
October University for Modern
Sciences and Arts (MSA), Egypt
HCI-LAB, Faculty of Computers
and Artificial Intelligence,
Helwan University, Egypt.
aezzat@msa.eun.eg

***Abstract*—This paper presents a comparative study of Machine learning and Deep learning models used in anomaly-based network intrusion detection systems. The paper has presented an overview of the previous work done in the field of ML and DL IDS, then an overview of the used datasets in reviewed literature was presented. Moreover, ML and DL models were tested on the KDD-99 dataset, and performance results were presented, compared, and discussed. Finally, areas of future research of critical importance are proposed by the authors.**

***Index Terms*—intrusion detection, machine learning, deep learning, comparative**

I. Introduction

In the past decade, dependence on network-attached devices has continuously grown. With such dependence, the importance of securing network-attached devices has grown in importance thus, network security has been an ever-evolving field. Among other interests, research in network security heavily aims to find methods to prevent malicious attacks to be executed on networks. Such attacks can cause disastrous outcomes organizations, industrial entities, governments, and individual privacy. Intrusion Detection Systems (IDS) are systems that aim to detect such malicious attacks, and are often coupled with Intrusion Prevention Systems (IPS) to prevent such attacks to bypass the implemented security measures and affect the inter-connected devices within networks. Therefore, Intrusion Detection Systems subjects of high demand and development interest. Intrusion Detection Systems are mainly divided into two main categories Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion detection Systems (HIDS), where NIDS are attached directly to the network's infrastructure fabric at critical network gateways, on the other hand HIDS are installed on individual hosts that are connected to the network. Within both IDS types, IDSs can detect malicious attacks using two main detection methods: Signature-based Detection (aka misuse detection) and Anomaly-based Detection [1] [2]. Signature-based detection works by detecting a specific signature which resembles a specific action set, while Anomaly-based detection works by differentiating between what is considered a normal packet or an abnormal one [1] [3]. Therefore, Signature-based detection is very useful in extremely stable environments which are not prone to change in potential attack methods, while Anomaly-based detection is a much more suitable method of detection of novel malicious attacks [1]. Therefore this paper will focus on using Anomaly-based detection as it suits real-world scenarios, where malicious attack signatures, are continuously changing. Securing the network against any potential attacks that might compromise the victim infrastructure is important.However to achieve that, conventional techniques becomes very weak when it comes to sophisticated attacks such as zero day attacks. This has led the research community to work on anomaly network detection. To prevent malicious use or accidental damage to the network's private data, its users, or their devices. Many proposed works have shed the light on using Machine learning techniques in network anomaly detection in order to distinguish between malicious and benign packets and use such techniques in the context of detection of abnormal packets in real networks [4]. Machine learning is a field in which computers are trained to predict the state of newly fed data based on adequate amounts of previously digested data given by humans as a verified fact. Computers can execute such operations using many different ML models, with each model usually having its advantages and can excel in different use-cases [4]. In the context of IDS, this paper reviews and compares many previous literature which use many different ML models aiming to find the most suitable model and methodology that can be used for effective Anomaly-based IDS.

This paper aims to present a comparative study of the reviewed existing literature in the field of machine learning and deep learning anomaly-based intrusion detection systems. Moreover, this paper tests a selected set of machine learning and deep learning models and illustrates a performance comparison between the utilized models.

II. Datasets

This section introduces different types of datasets which were used in other previous works and in this work. However, to the best of our knowledge and our research we found only these datasets which showed some drawbacks during our experiments due to the age of creation. Such drawbacks are further discussed by Divekar et al. [5], Sung et al. [6], and Janarthanan and Zargari [7].

A. KDD-99 Dataset

KDD-99 dataset has been released on 1999 and is based on a 7 week period of packet capture exports made by DARPA's IDS evaluation program in 1998. The KDD-99 dataset is composed of nearly 4.9 million records of network packets, some of which are normal packets and others which are malicious attack packets. Each packet is listed based on 41 features and a target which identifies whether the packet is a normal/attack packet. The dataset targets 5 different attack classes, which break down to 22 different attack subclasses [4]. The dataset is available as a complete version containing 4.9 million records or a 10% version as a less resource intensive alternative. The KDD-99 dataset, despite having some inherent problems such as uneven data, outdated attacks, redundant data, and high skewedness as discussed in [5], is still the most widely used dataset for ML IDS training and performance evaluation [8].

B. NSL-KDD Dataset

The NSL-KDD is an iteration over the KDD-99 dataset. The NSL-KDD dataset utilizes the same feature and target set, however, it aims to remove the inherent problems within the KDD-99 dataset. Some of which are the removal of redundant and duplicate records in order to avoid bias due to frequently occurring records, providing less records while maintaining relevance to improve performance on less resourceful training and testing environments, and selection of records within each difficulty level is based on an inversely-proportional relationship to the percentage of records in the KDD-99 dataset. Such improvements enable a more accurate representation of the performance of each ML model. [2]

C. ISCX IDS Dataset

The ISCX dataset is composed of nearly 2 million packets that have been collected over a 7-day period. The packets are described with 20 features and the target recognizes the presence of 4 different attack types which are: Brute Force SSH, Infiltrating, HTTP DoS, and DDoS which compose 2% of the whole dataset packet count [9]. However, ISCX dataset is not only a very recent dataset with fewer maturity time compared to KDD-99 dataset, it also has suffers from some weaknesses. To elaborate, ISCX dataset application protocols lack strong classification. To add, ISCX relies on human interferance to write scripts that describe attack scenarios, which significantly decrease the usability of the dataset in a real-world scenario. [10]

D. UNSW-NB15

The UNSW-NB15 dataset has been developed by Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). The dataset has been extracted and derived from 100GBs of raw pcap data using the IXIA PerfectStorm tool, and recognizes nine attack types being: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The dataset labels each vector based on 49 features with the target label. The UNSW-NB15 dataset is considered by some to be a proper advocate as an alternative for the popular KDD-99 dataset [5]. However, the UNSW-NB15 has shortcomings; for example, the dataset contains several features that are redundant and also affect the utilized models' accuracy [6] [7]. To add due to the dataset being relatively new [7], not much work has been done utilizing the dataset, therefore using the dataset as a benchmark for the tested models might lead to misleading results.

III. Related work

The related work of ML and DL in anomaly-based IDS will be reviewed and illustrated in the chronological order of ML and DL pipelines as seen in the reviewed literature, which is in the following order: Data collection, Pre-processing, and Classification [11]. Figure 1 illustrates a ML and DL IDS pipeline structure commonly used in reviewed literature discussed in this section.

A. Data Collection

Despite some inherent problems in the KDD-99 dataset, it is considered the default bench marking standard in ML and DL IDS performance evaluation [12]. It has been found that most of the reviewed literature utilizes the KDD-99 dataset in the data collection phase such as in [8] [13] [14] [3]. A closely-popular choice is the NSL-KDD dataset, which has been utilized in [2] [15] [16]. Moreover, other datasets has been used such as the ISCX dataset in [9] and UNSW-NB15 dataset as used in [5] [12]

B. Pre-processing

In the previous reviewed literature, it has been found that the pre-processing pipeline mainly consists of two stages: numericalization of categorical features, and standardization of highly variant data. To elaborate, the KDD-99 dataset has three categorical features: protocol_type, service and flag [4], all of which need to be labeled as numerical representations of such categorical features [2] [8]. In the reviewed literature, it has been observed consistent popularity of usage of One-Hot Encoders and Label Encoders in the numericalization of the categorical features in the used datasets such as in: [17] [3]. However, other methods of encoding has been used such as the proposition of a novel Non-Symmetric Deep Auto-Encoder

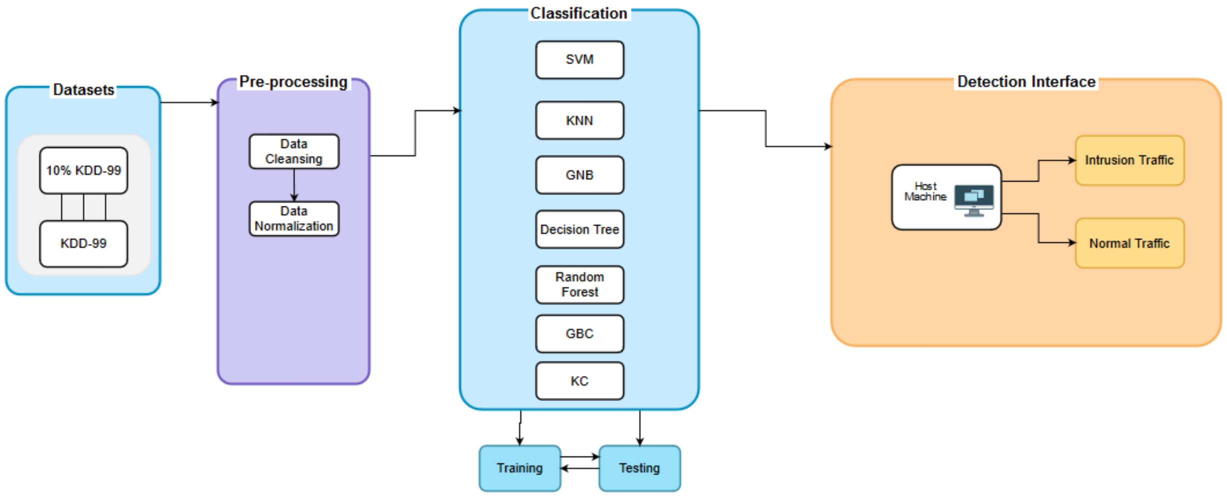


Fig. 1. ML and DL IDS pipeline structure commonly used in literature

(NDAE) in [8]. Moreover, in the next stage of pre-processing, standardization of highly variant data aims to scale data into numerically similar range around the mean of the distribution, and scaling by the standard deviation value. Standardization of highly variant features is a common practice among reviewed literature as observed in [5] [18], the authors of this paper believe that the need of standardization originates from the flexible nature of network usage spikes, which reflects onto packet features being highly variant.

C. Classification

This section illustrates the usage of different classification methods in the reviewed related literature.

Mainly, the utilization of ML classifiers and DL classifiers has been observed. Moreover, such observations will inspire this paper to further test a select set of highly-performing classifiers to be further tested and compared on a local testing environment [19].

1) Support Vector Machines (SVM): Support vector machines are one of the machine learning techniques, based on training vectors. With high-dimensional feature space plotting and labeling each vector by its class. SVM usually used for classification and regression purposes. Data can be classified by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in feature space. It transform dataset into a set of transformations, to identify the optimal margin point between the sets of data based on the selected features of the dataset. Although SVMs were originally published in 1995 they are still considered one of the most robust machine learning algorithms up-to-date [20] [21].

Previous literature has utilized SVMs in the field of ML IDS to classify the benign and malicious network activities through packet inspection. For instance, Divekar,at.al [5] developed Machine learning model to detect anomaly based network intrusion activities using KDD-99, NSL-KDD, and UNSW-NB15 datasets, respectively. The model has been tested and achieved weighted f1-scores of 91, 82, 82.

2) K-Nearest Neighbors (KNN): KNN is a non-parametric machine learning algorithm. Likewise SVM, KNN can be used for both, classification and regression purposes. To achieve that the k-closes training model requires an input in the feature space. To classify a set of vectors in the dataset into k classes, which is then used to predict the class of a new vector.Many of respected research efforts have utilized this algorithm in an attempt to detecting network attacks. Such as [22] the authors deployed 5-closest neighbors and achieved average accuracy of 97.92%, 99.43%, 93.44%, 99.63%, and 99.19% of detecting the norma, dos, probe, u2r, and r2l classes of the KDD-99 dataset.

3) Decision Trees (DT): Decision tree Algorithm is constructed of decision and leaf nodes. Each decision node relates to a test X over a single attribute of the input data and has a number of branches, each of which handles an outcome of the test X. Each class is represented by a leaf node that is the result of decision for a case [21].

Zainal et al. [23] proposed a Decision Tree and and Genetic model based on machine learning IDS. Moreover, the study utilized the KDD-99 dataset and applied the proposed machine learning model and was able to optimize then accuracy scores compared to Linear Genetic Programming, Adaptive Neural Fuzzy Inference System,and Random Forest classifiers.

4) Ensemble Classifiers: Ensemble Models rely on utilizing multiple machine learning models together in the aim of achieving higher predictive performance. Ensemble models are able to combine the predictions of a finite set of multiple machine learning models, therefore compensating the accuracy shortcoming of each individual model. [24]

Hamid et al has utilized the AdaBoost (Adaptive Boosting) ensemble model in the context of ML IDS. Their research was conducted using Waikato Environment for Knowledge Analysis (WEKA) and utilized the KDD-99 dataset. Moreover, the model was able to achieve 97.859% average accuracy over the 5 KDD-99 classes [4].

Furthermore, another ensemble model is the Random Forest classifier, which generates multiple trained decision trees and selects the decision tree of the mean prediction value of all decision trees that have been generated [25]. Divekar et al. [5] also utilized WEKA's Random Forest classifier in ML IDS context, with a number of trees were cross-validated on up to a maximum of 100. The model was able to achieve accuracy scores of 94%, 81%, and 88.5% on KDD-99, NSL-KDD, and UNSW-NB15 datasets, respectively.

5) Naive Bayes (NB): Naive Bayes Classifiers are simple probabilistic classifiers which utilize the statistical bayes theorem alongside a naive assumption of feature independence. Naive Bayes classifiers return probabilities of target occurrence, as they are generative models. [26] Divekar et al. [5] and Hamid et al. [4] have both utilized Naive Bayes Classifiers in the context of machine learning IDS. Divekar et al. were able to achieve average accuracy scores of 86%, 78%, and 76% on the KDD-99, NSL-KDD, and UNSW-NB15 datasets, respectively.

6) Deep Neural Networks (DNN): [12] Deep Neural Networks are Artificial Neural Networks with multiple layers between the input layer and the output layer. Such layers are simply mathematical manipulation of the data passed onto the layer. Consequently, DNNs considered as feedforward networks, meaning that the data is fed to each layer on an individual basis and then forwarded to the layer below. [27]

Tang et al. [28] proposed a deep neural network anomaly-based IDS that utilized the NSL-KDD dataset. The DNN model was constructed of an input layer, 3 hidden layers, and an output layer; moreover, the model achieved an average accuracy of 75.57% over 4 (dos, r2l, u2r, probe) classes of the NSL-KDD 5 attack classes.

IV. Experiment

A review of the common promising ML and DL models from the previously reviewed literature will be recreated, analyzed, and discussed in detailed. Moreover, the most suitable select set of models will be indicated.

The experiment was done on a local testing bench with the following specifications: AMD Ryzen 3600 6 Core, 12 Thread CPU, 16GB 3200Mhz CL16 Memory, AMD Radeon RX 570 8GB GPU, Crucial MX300 256GB Sata SSD, Windows 10 Pro (v.1909), Anaconda3(1.9.12) w/ JupyterLab(1.1.4).

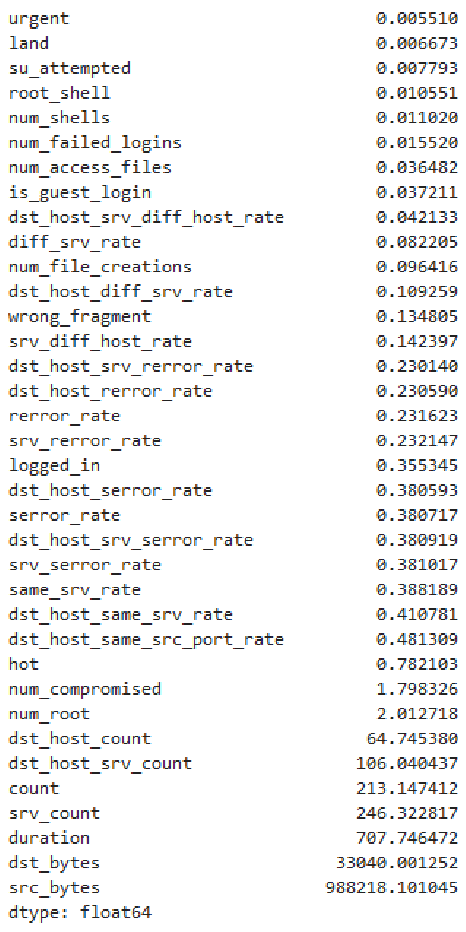
A. Data Collection

As illustrated by Shone et al. [8], the popularity of the KDD-99 makes it considered by some to be the de facto standard of IDS ML and DL testing datasets. Therefore the authors of this paper decided to test the use of the select ML and DL models on the commonly-used KDD-99 dataset in practice which can reflect the findings relative to a common benchmark.

The KDD-99 dataset is publicly available on University of California Irvine Knowledge Discovery in Databases Archive. A comparison of both the 10% and full corrected versions of the dataset will be denoted, therefore, both dataset versions will be attained in the data collection stage. Both versions of the KDD-99 datasets will be utilized, as it was illustrated in I the larger version of the dataset scales up the frequency of the common classes while keeping unchanged frequencies on the least 2 common classes. Therefore, it is of critical importance to test how various models perform in an environment which resembles high skewness in common class occurrence frequency.

B. Pre-processing

1) Data cleansing: Data cleansing is performed on the dataset, as the presence of non-cleansed data can cause undesirable effects on the final models' results. First, the authors of this paper verified the inexistence of null values within both dataset versions, and no nulls were found in both corrected KDD-99 dataset versions. To add, due to the high variance in the dataset features, as shown in figure 2, standardization of highly variant data was performed on both datasets to improve classifier performance during the classification stage.



```
urgent                              0.005510
land                                0.006673
su_attempted                        0.007793
root_shell                          0.010551
num_shells                          0.011020
num_failed_logins                   0.015520
num_access_files                    0.036482
is_guest_login                      0.037211
dst_host_srv_diff_host_rate         0.042133
diff_srv_rate                       0.082205
num_file_creations                  0.096416
dst_host_diff_srv_rate              0.109259
wrong_fragment                      0.134805
srv_diff_host_rate                  0.142397
dst_host_srv_rerror_rate            0.230140
dst_host_rerror_rate                0.230590
rerror_rate                         0.231623
srv_rerror_rate                     0.232147
logged_in                           0.355345
dst_host_serror_rate                0.380593
serror_rate                         0.380717
dst_host_srv_serror_rate            0.380919
srv_serror_rate                     0.381017
same_srv_rate                       0.388189
dst_host_same_srv_rate              0.410781
dst_host_same_src_port_rate         0.481309
hot                                 0.782103
num_compromised                     1.798326
num_root                            2.012718
dst_host_count                     64.745380
dst_host_srv_count                106.040437
count                             213.147412
srv_count                         246.322817
duration                          707.746472
dst_bytes                       33040.001252
src_bytes                      988218.101045
dtype: float64
```

Fig. 2. standard deviation of each feature in 10% KDD-99

2) Analysis and Visualization: General analytics of the dataset is performed for visualization and further understanding of the dataset construction.

TABLE I
ATTACK CLASSES AND FREQUENCY OF OCCURRENCE IN 10% AND FULL DATASET

	10% KDD-99 Dataset	Full KDD-99 Dataset
Dos	391458	3883370
Normal	97278	972781
Probe	4107	41102
R2L	1126	1126
U2R	52	52

Table (I) illustrates each attack class and the frequency of its occurrence in both dataset versions is observed. This outcome is of critical importance as the high variation in class frequency can cause high detection difficulty in less common classes.

3) Feature selection: 3 illustrates a correlation heatmap of the dataset's features is display, which shall be used to identify extremely-correlated features which can be dropped. Since the authors have utilized the corrected version of the dataset, no extremely correlated features have been identified. However, the heatmap still is useful for further analysis of the dataset as it identifies features which are highly-uncorrelated and can affect the model's accuracy positively.

Moreover, there were 2 features which had less than 2 occurrences of unique values, which were the 'is_host_login' and 'num_outbound_cmds', therefore such features were considered redundant features and have been dropped to further decrease unnecessary performance overhead on the models.

4) Label encoding: In the label encoding stage, three categorical features 'protocol_type', 'flag', and 'service' were found and scikit-learn Label Encoding was utilized to encode those features' values into numerical representations.

5) Scaling and Splitting Dataset: Scaling has been done using the Min Max Scaler and splitting of the data is done using the scikit-learn python library. The dataset was split into training and testing datasets of a test-to-train ratio of 0.2.

C. Classification

A select choice of models which showed promising results in reviewed literature will be tested. The authors will utilize scikit-learn library to perform the models previewed within subsections (1-7), and the Keras classifier library will be utilized in subsection (8). Moreover, scores and metrics of all models will be illustrated on the basis of precision, recall, f1-score, and support metrics. A detailed descriptions of the metrics is presented below:

Precision:
The average precision (AP) scores are computed as a value between 0 and 1, with 0 referring to a model with 0% accuracy and 1 being a model achieving 100% accuracy. AP is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

where TP is the number of positive items flagged as positive by the model and FP is the number of negative items flagged as positive by the model.

Recall:
Recall, also known as Sensitivity or True Positive Rate (TPR), calculates the ratio of all correctly detected vectors within the data to all vectors that should be detected in an ideal case.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

where TP and FN are True Positives and False Negatives, respectively. True positives being the items that were correctly detected by the model, and false negatives being positive items that were not detected by the model.

F1-score:
F1-score is the harmonic mean of both, precision and recall, and reflects a score of the model's accuracy.

$$F1 = \frac{2 * TP}{2 * TP + FN + FP} \tag{3}$$

where R_n and P_n are the precision and recall at the nth threshold. With random predictions, the AP is the fraction of positive samples.

Support:
Support is the number of occurrences found of each class of items in the dataset.

1) Support Vector Machines (SVM): SVM is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 66 seconds and testing time was 6.5 seconds, with both the training and testing scores being 0.99. On the full KDD-99 dataset the training time was 2472 seconds and testing time was 202 seconds, with both the training and testing scores being 0.99.

2) K-Nearest Neighbors (KNN): The KNN model is tested on both the 10% version of the dataset. KNN wasn't utilized on the full version of the dataset as it was expected to show extremely high training time, and a trial test confirmed the authors' expectations.On the 10% version of the dataset, the training time was 263 seconds and testing time was 226 seconds, with both the training and testing scores being 0.99.

3) Gaussian Naive Bayes (GNB): The Gaussian Naive Bayes model is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 0.5 seconds and testing time was 0.38 seconds, with both the training and testing scores being 0.88. On the full KDD-99 dataset the training time was 5.39 seconds and testing time was 3.41 seconds, with both the training and testing scores being 0.88.

4) Decision Tree (DT): DT model is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 1.1 seconds and testing time was 0.02 seconds, with both the training and testing scores being 0.99. On the full

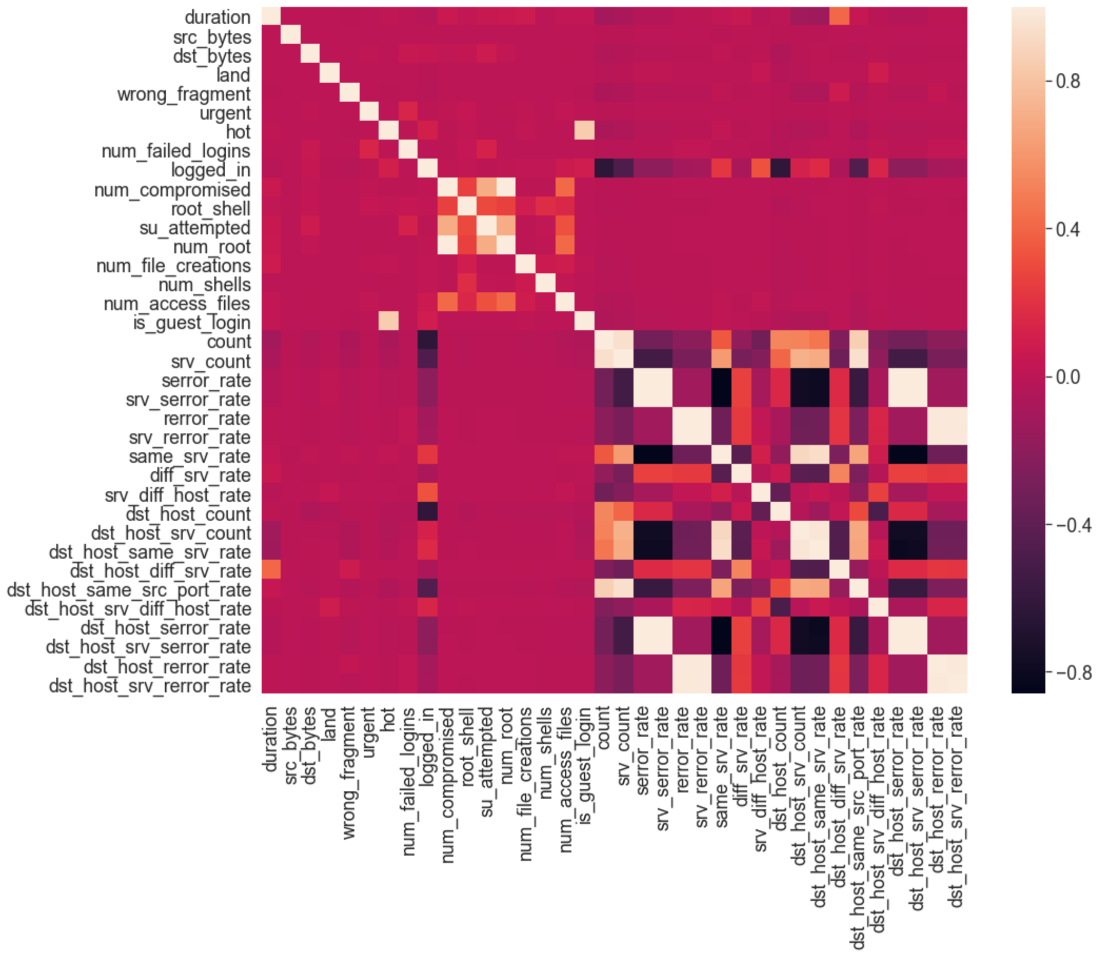


Fig. 3. 10% KDD-99 features correlation heatmap

KDD-99 dataset the training time was 15.8 seconds and testing time was 0.64 seconds, with both the training and testing scores being 0.99.

5) Random Forest (RF): RF model is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 7.11 seconds and testing time was 0.39 seconds, with both the training and testing scores being 0.99. On the full KDD-99 dataset the training time was 119 seconds and testing time was 4 seconds, with both the training and testing scores being 0.99.

6) Logistic Regression (LR): LR is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 12.73 seconds and testing time was 0.02 seconds, with both the training and testing scores being 0.99. On the full KDD-99 dataset the training time was 146.9 seconds and testing time was 0.21 seconds, with both the training and testing scores being 0.99.

7) Gradient Boosting Classifier (GBC): GBC is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 353 seconds and testing time was 1.2 seconds, with both the training and testing scores being 0.99. On the full KDD-99 dataset the training time was 4019 seconds and testing time was 10.1 seconds, with both the training and testing scores of the 10% dataset being 0.987 and the full dataset being 0.99.

8) Keras Classifier (KC): KC is tested on both the 10% version and the full version of the KDD-99 dataset, respectively. On the 10% version of the dataset, the training time was 300 seconds and testing time was 0.5 seconds, with both the training and testing scores being 0.99. On the full KDD-99 dataset the training time was 2900 seconds and testing time was 4.3 seconds, with both the training and testing scores being 0.99.

V. Evaluation and Results

As illustrated in II, the various models were utilized and the performance was measured based on precision, recall, f1-score, and support. Performance was tested on both the 10% and full versions of the KDD-99 datasets.

TABLE II
RESULTS OF ALL TESTED MODELS ON BOTH THE 10% AND FULL VERSION OF KDD-99 DATASET

		10% KDD				Full KDD			
		Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
SVM	dos	1.00	1.00	1.00	78439	1.00	1.00	1.00	776980
	normal	1.00	1.00	1.00	19260	1.00	1.00	1.00	194138
	probe	1.00	0.98	0.99	872	1.00	0.98	0.99	8311
	r2l	0.93	0.91	0.92	223	0.87	0.75	0.81	247
	u2r	1.00	0.18	0.31	11	1.00	0.18	0.31	11
KNN	dos	1.00	1.00	1.00	78439	-	-	-	-
	normal	1.00	1.00	1.00	19260	-	-	-	-
	probe	1.00	0.99	0.99	872	-		-	-
	r2l	0.95	0.96	0.95	223	-	-	-	-
	u2r	0.83	0.45	0.59	11	-	-		-
GNB	dos	0.98	0.94	0.96	129106	0.98	0.95	0.96	1281483
	normal	0.97	0.64	0.77	32167	1.00	0.62	0.76	321021
	probe	0.09	0.99	0.17	1348	0.10	0.97	0.18	13594
	r2l	0.29	0.39	0.33	387	0.03	0.44	0.06	365
	u2r	0.01	0.74	0.01	19	0.00	0.90	0.00	20
DT	dos	1.00	1.00	1.00	129106	1.00	1.00	1.00	1281483
	normal	0.98	0.98	0.98	32167	0.98	1.00	0.99	321021
	probe	0.55	0.90	0.68	1348	0.99	0.64	0.78	13594
	r2l	0.00	0.00	0.00	387	0.00	0.00	0.00	365
	u2r	0.00	0.00	0.00	19	0.00	0.00	0.00	20
RF	dos	1.00	1.00	1.00	129106	1.00	1.00	1.00	1281483
	normal	1.00	1.00	1.00	32167	1.00	1.00	1.00	321021
	probe	1.00	0.99	0.99	1348	1.00	0.99	1.00	13594
	r2l	0.98	0.96	0.97	387	0.96	0.93	0.94	365
	u2r	0.93	0.68	0.79	19	1.00	0.20	0.33	20
LR	dos	1.00	1.00	1.00	129106	1.00	1.00	1.00	1281483
	normal	0.98	1.00	0.99	32167	0.99	1.00	1.00	321021
	probe	0.98	0.90	0.94	1348	0.98	0.90	0.94	13594
	r2l	0.84	0.82	0.83	387	0.14	0.02	0.03	365
	u2r	0.86	0.32	0.46	19	0.50	0.05	0.09	20
GBC	dos	1.00	1.00	1.00	129106	1.00	1.00	1.00	1281483
	normal	1.00	0.95	0.97	32167	1.00	1.00	1.00	321021
	probe	1.00	0.65	0.78	1348	1.00	0.98	0.99	13594
	r2l	0.13	0.77	0.23	387	0.70	0.66	0.68	365
	u2r	0.28	0.79	0.42	19	0.64	0.35	0.45	20
KC	dos	1.00	1.00	1.00	129106	1.00	1.00	1.00	1281483
	normal	1.00	1.00	1.00	32167	1.00	1.00	1.00	321021
	probe	0.99	0.98	0.99	1348	0.97	0.99	0.98	13594
	r2l	0.86	0.89	0.87	387	0.00	0.00	0.00	365
	u2r	0.00	0.00	0.00	19	0.00	0.00	0.00	20

VI. Discussion

SVM and KNN models showed promising accuracy scores in the less common classes 'r2l' and 'u2r' in a comparison with other respected previous work results that fell behind in the stated less common classes such as GNB, DT, and GBC. However, both models fell behind significantly in terms of training and testing times, which rises concerns whether SVM and KNN can be deployed in real time detection considering high bandwidth and time-critical networks. Moreover, the KNN model was unable to finish training during a 24-hour period, therefore the test was aborted and KNN was considered unsuitable for real-world usage.

Furthermore, LR showed satisfactory accuracy scores, excellent training time, and outstanding training time in the smaller dataset, however, the model fell behind significantly in the training time and accuracy of the larger dataset in the less common classes; therefore, it shall further be investigated whether the model can be used in a real time detection on a non-time critical environments or whether the model can be utilized in an offline-learning ML IDS.

The GBC model showed adequate accuracy scores in the common classes; however it fell behind significantly in the less common classes showed unsatisfactory training and testing time in both datasets.

Moreover, GNB and DT were able to achieve excellent training and testing time in both versions of the dataset,however, the shortcomings of both models in an IDS context is clear due to the overall low average accuracy scores achieved in the less common attack classes. The KC model was able to achieve satisfactory results in the common attack classes, however it fell behind significantly in training times of both datasets and was unable to detect any attacks of both less common attack classes in the larger datasets, which shows that this model is unsatisfactory for use in a real-world scenario.

Finally, the RF model showed outstanding accuracy in all 5 classes, while achieving satisfactory training and testing times. Despite falling behind in training time of the larger dataset, RF was able to achieve extremely satisfactory results even in both dataset versions, which was remarkable considering the significant accuracy drop other models suffered from when being tested on the full dataset. Therefore, the RF model's results were considered extremely promising and further testing of the model is highly recommended.

VII. Conclusion and Future Work

This paper presents a comparative study of ML and DL models used in anomaly-based network intrusion detection systems. The paper has presented an overview of the previous work done in the field of ML and DL IDS, then an overview of the used datasets in reviewed literature was presented. Moreover, ML and DL models were tested on the KDD-99 dataset, and performance results were presented and compared. Various models showed specific advantages and disadvantages and no specific model was considered completely superior over other models, however, the RF model showed promising results and shall further be tested in a real-world IDS scenario.

Furthermore, it should be noted that the field of ML and DL IDS is relatively new and further research is extremely needed. Therefore this paper lay the ground for many achievements as in the future work to be done especially in the areas of online learning ML and DL IDS, as it represent a more realistic utilization of ML and DL in IDSs. To add, another recommended field of research is improving current datasets, as the datasets available showed inherent problems such as being dated and unrepresentative of modern network attacks. Moreover based on the fore mentioned dataset limitations in this field, which attracted our attention in the future work. Finally, another important area of research is targeting mobile-specific network attacks, either by providing specialized mobile IDS network datasets or providing mobile-specific ML and DL IDS architectures.

References

[1] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, p. 227–261, Nov. 2000. [Online]. Available: https://doi.org/10.1145/382912.382914

[2] B. Ingre and A. Yadav, "Performance analysis of nsl-kdd dataset using ann," in *2015 International Conference on Signal Processing and Communication Engineering Systems*, 2015, pp. 92–96.

[3] W. Hu and W. Hu, "Network-based intrusion detection using adaboost algorithm," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI '05. USA: IEEE Computer Society, 2005, p. 712–717. [Online]. Available: https://doi.org/10.1109/WI.2005.107

[4] Y. Hamid, M. Sugumaran, and L. Journaux, "Machine learning techniques for intrusion detection: A comparative analysis," in *Proceedings of the International Conference on Informatics and Analytics*, ser. ICIA-16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2980258.2980378

[5] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, pp. 1–8.

[6] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *2003 Symposium on Applications and the Internet, 2003. Proceedings.*, 2003, pp. 209–216.

[7] T. Janarthanan and S. Zargari, "Feature selection in unsw-nb15 and kddcup'99 datasets," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1881–1886.

[8] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[9] S. Soheily-Khah, P. Marteau, and N. Béchet, "Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the iscx dataset," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018, pp. 219–226.

[10] Y. Lin, P. Lin, S. Wang, I. Chen, and Y. Lai, "Pcaplib: A system of extracting, classifying, and anonymizing real packet traces," *IEEE Systems Journal*, vol. 10, no. 2, pp. 520–531, 2016.

[11] G. Karatas, O. Demir, and O. Koray Sahingoz, "Deep learning in intrusion detection systems," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 2018, pp. 113–116.

[12] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[13] G. Meena and R. R. Choudhary, "A review paper on ids classification using kdd 99 and nsl kdd dataset in weka," in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, 2017, pp. 553–558.

[14] Z. Muda, W. Yassin, M. N. Sulaiman, and N. I. Udzir, "Intrusion detection based on k-means clustering and naïve bayes classification," in *2011 7th International Conference on Information Technology in Asia*, 2011, pp. 1–6.

[15] L. Hakim, R. Fatma, and Novriandi, "Influence analysis of feature selection to network intrusion detection system performance using nsl-kdd dataset," in *2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*, 2019, pp. 217–220.

[16] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *2015 National Aerospace and Electronics Conference (NAECON)*, 2015, pp. 339–344.

[17] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *2014 Second International Conference on Advanced Cloud and Big Data*, 2014, pp. 247–252.

[18] H. Nkiama, S. Zainudeen, and M. Saidu, "A subset feature elimination mechanism for intrusion detection system," *International Journal of Advanced Computer Science and Applications*, vol. 7, 04 2016.

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[20] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273–297.

[21] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, vol. 2, 2002, pp. 1702–1707 vol.2.

[22] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary pso and k-nearest neighborhood algorithm," in *2017 11th International Conference on Information Communication Technology and System (ICTS)*, 2017, pp. 181–186.

[23] A. Zainal, M. Maarof, and S. M. Shamsuddin, "Ensemble classifiers for network intrusion detection system," *Journal of Information Assurance and Security*, vol. 4, pp. 217–225, 07 2009.

[24] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Int. Res.*, vol. 11, no. 1, p. 169–198, Jul. 1999.

[25] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, p. 832–844, Aug. 1998. [Online]. Available: https://doi.org/10.1109/34.709601

[26] H. Zhang, "The optimality of naive bayes," vol. 2, 01 2004.

[27] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, p. 85–117, Jan 2015. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2014.09.003

[28] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263.