EVALUATING THE ACCURACY AND PREDICTION TIME OF DIFFERENT MACHINE LEARNING TECHNIQUES FOR NETWORK INTRUSION DETECTION SYSTEMS

Zein Hajj-Ali - 101020677   Nhat Hieu Le - 101124254

I. Abstract

It has been shown that the traditional Network Intrusion Detection System (NIDS) has imposed limitations: Zero-day exploitation, high False Alarm Rate (FAR) and inability to process encrypted packets. Recently, machine learning (ML) and deep learning (DL) techniques have become promising alternative approaches to overcome these aforementioned disadvantages. This paper investigates the findings of research papers working on NIDS using DL algorithms and compares the accuracies of the proposed DL solutions with popular ML alternatives such as: Naive Bayes, Random Forest, Bayes Network, etc. The prediction time it takes to classify a record of network traffic will also be compared, since efficiency in time is an important metric for the NIDS as malicious activities need to be predicted and dealt with as soon as possible. The UNSW-NB15 dataset used in this paper is one that is widely studied for the purposes of anomaly detection. The dataset includes 9 different attack types for anomalous records, and so a few multi-label classification methods are also examined. Additionally, it will be shown that given a large enough dataset, deep learning methods result in a higher accuracy and competitive prediction time.

Key words: Deep Learning, Machine Learning, NIDS, UNSW-NB15 dataset

II. Introduction

Cybersecurity has become a real threat for businesses. There has been an increase in cybercrime targeting large corporations in recent years. It has been reported that data breaches had exposed over 4.1 billion records in the first half of 2019 and that the damage caused by online crime is projected to be $6 trillion annually by 2021 [1]. Furthermore, heterogeneous types of malicious traffic such as zero-day attacks, social engineering, denial of service attacks (DoS) etc. have increased at an exponential rate [2]. Unsurprisingly, cybercrime is now considered a major issue for corporations in the digital era that need to maintain the continuous availability of their services to their customers and avoid damaging their reputation. A Network Intrusion Detection System keeps an eye on network traffic within companies to study the normal traffic flow and detect any anomalous behavior. The longer it takes to identify an intrusion, the more damage the companies will incur. Therefore, monitoring corporations' high volume network traffic and uncovering possible intrusions and malicious activity in a timely manner are ultimate goals of NIDS. Moreover, being able to distinguish the kind of abnormal activities accurately and executing the appropriate neutralization processes are key metrics for the NIDS system. Depending on the methods used, there are always tradeoffs between the accuracy of the NIDS and its speed.

Due to the limitations of conventional NIDS such as high false alarm rates, incapable of identifying zero-day exploits etc, several research papers have already delved into machine learning as an alternative solution and have compared conventional ML techniques to each other as in [3]. This paper also considers ML techniques for intrusion detection on UNSW-NB15 dataset. The main contributions of this research are: firstly, to determine the most efficient algorithms for an intrusion detection system when comparing conventional machine learning techniques like J48, REPTree, KNN and Random Tree to deep learning algorithms on both the partitions of UNSW-NB15 dataset and on the full dataset (discussed later) for both intrusion detection and malicious activities classification. Secondly, the prediction times of the different methods when analysing unseen records will also be a metric to analyze and evaluate the efficiency of the algorithms used in NIDS. Faster training time and increasing accuracy are the main purposes of employing feature selection methods.

The remainder of the project paper will be organized as follows. Section III will provide fundamental knowledge about machine learning algorithms and deep learning techniques, and tools such as Weka, Tensor Flows, Scikit-learn will be introduced as well. Section IV will give the literature review relating to previous work that will be used in this project. Section V will describe details about UNSW-NB15 dataset and how data preparation is conducted. Experiments and methodology will be discussed in Section VI. Finally, results and conclusions will be analyzed in Section VII and Section VIII respectively.

III. Background

Machine learning is a set of methods that are able to detect patterns in data and then use the uncovered patterns to predict future data, predict the class of a piece of unseen data, or conduct other types of decision making under uncertainty [4]. In general, it can be categorized into unsupervised learning, which looks for patterns within the given samples, supervised learning, which predicts the mapping between input features and outputs, and reinforcement learning, which behaves based on reward and punishment signals [4]. The supervised learning approach will be used to solve the prediction problem of detecting anomalous network traffic. Several machine learning algorithms use different approaches to identify the relation between input features and output goals.

Below are some of the software tools, libraries used and machine learning algorithms used for this project:

TensorFlow: an open-source software library written in Python, C++, and CUDA that is developed by Google. Although it can be used for various range of machine learning tasks, TensorFlow has a focus on training and inference of Deep Neural Networks. Therefore, this library is used to implement the deep learning algorithms in this project.

Scikit-Learn: an open-source software machine learning library for Python programmers. In addition to myriad support of classification, clustering, and regression algorithms, Scikit-Learn integrates well with popular numerical and scientific libraries like Pandas and Numpy whilst providing efficient data transformers libraries. As a result, this library is leveraged for data cleaning, scaling, and feature transformation.

Numpy and Pandas: both are some of the most popular Python libraries which are widely used for data manipulation when interacting with a multi-dimensional dataset. In this project, both Numpy and Pandas are used for data preparation and implementing deep learning algorithms.

Weka: an open-source machine learning software developed by the University of Waikato and written in Java. It offers access to either graphical user interface or Java command line and has become more and more popular with researchers since it supports a large number of standard machine learning tools [5]. In this project, Weka will be used to run data preprocessing and below machine learning algorithms.

K-Nearest Neighbors: the algorithm looks for K points/neighbors, in the training set which are nearest to the input record and assigns the label of the dominant class in that neighborhood [4]. Therefore, the number of neighbors are usually odd. K-NN is called “IBK” in Weka.

Random Tree: it applies a bagging method to generate a random set of data for forming the decision tree. For conventional trees, each node is separated by the variable that provides the most information gain among all attributes while this algorithm chooses a random subset of attributes at each node [6]. It is also implemented in Weka under the same name.

REPTree - Reduced Error Pruning Tree: the algorithm is a fast decision tree learner that uses information gain as the splitting decision and prunes using reduced error pruning available in Weka as well [6].

Naive Bayes Classifier: it is based on Bayesian probability theorem with the assumption that all features are conditionally independent given a class label [4]. Despite its simplicity, this algorithm outperforms other alternative approaches in terms of the need for computation resources and faster training and building time.

J48: it is the implementation of the C4.5 algorithm found in Weka, an extension of the famous ID3 algorithm, to generate a decision tree-based classifier. It is also known as a statistical classifier.

Additionally, algorithms employing *deep learning*, a subset of machine learning including algorithms inspired by the function and structure of the brain called artificial neural networks, will also be built and evaluated in this project. deep learning differentiates itself from the classical approach by generating the set models which consists of many successive transformation layers of data chained from top to bottom. Thus it can be referred to as multi-level representation learning [7].

IV. Literature Review

Traditional NIDS have been hindered by high FAR and an incapablity to detect zero-day exploitations. To this extent, a multitude of researchers have recently tried to use machine learning to solve the intrusion detection problem on the UNSW-NB15 dataset in particular.

Nawir M, et al. in [8] proposed a binary classification on the partition UNSW-NB15 dataset using Weka. The authors applied tenfold cross validation for evaluating the performance of Average One Dependence Estimator, Bayesian Network and Naive Bayes algorithms and 43 features are all used except for “id” and “attack_cat” . Accuracy and time to build the model are the metrics used to evaluate the algorithms. This paper is limited in that the author neglects the features selection process as including highly correlated features may induce overfitting.

Anurag Das, et al. in [3] investigated the anomaly detection and attack type classification of nine machine learning techniques. Additionally, the effects of nine feature selection algorithms are also experimented and evaluated. The authors conducted three experiments. First, the accuracy of the nine machine learning algorithms for intrusion detection are evaluated. Second, 81 mini experiments are performed by combining nine machine learning algorithms with nine feature selection techniques. Both of the experiments are conducted on the partitioned UNSW-NB15 dataset using tenfold cross validation as an evaluation method. Third, seven machine learning algorithms are assessed on their capability of performing a multi-class classification of the type of attack on the BoT-IoT dataset when selecting only the ten best features. Similar to the previous paper, the authors consider accuracy and time to build as the metrics for comparison as well. However, this paper does not include deep learning methods for comparison and the multi-class classification is run on a different dataset. Implementation is done using Weka.

Alin F., et al. in [9] proposed a two level hierarchical system where anomaly detection is performed first then multi-label classification is executed successively. Four datasets (UNSW-NB15, KDD 99, NSL-KDD, CIC-IDS-2017) are used for training and testing classical machine learning algorithms: Naive Bayes, K-NN, SVM, Random Forests, Multilayer Perceptron Neural Networks and Convolution Neural Network. Accuracy is the key metric for evaluating the performance of the algorithms. However, this paper does not explicitly mention how the validation method is conducted and the time to build the model is not considered either. In addition, the authors skipped feature selection steps.

Muna, et al. in [10] introduced a deep learning method for anomaly detection for Internet Industrial Control Systems that uses an autoencoder for feature extraction of normal traffic and a deep learning feed forward neural network for binary classification. The authors work on the partitioned UNSW-NB15 and full NSL-KDD datasets. However, they only compared the accuracy of the proposed method with that of classical machine learning methods when using the NSL-KDD dataset.

M. Al-Zewairi, et al. in [11] proposed a deep learning method based on artificial neural networks using back-propagation and stochastic gradient descent. The method is applied for the binomial classification for NIDS on the full UNSW-NB15 full data set. Additionally, the Gedeon method is applied to select the top 15% of important features. Accuracy and FAR are the metrics used for evaluation.

In this paper our purpose is to produce a comprehensive comparison between classical machine learning algorithms and deep learning methods. Not only will the accuracy be taken into account, but the time to build and prediction time are also considered.

V. Data Preparation

In this section, we will introduce the data set that our methods will be evaluated on. We begin by describing how the dataset is constructed, after which, data cleaning and data manipulation steps are discussed. Finally, we will end this section by discussing the details of transforming and preparing features. Mustafa et al. in [12] generated the UNSW-NB15 at UNSW Canberra. The authors employed IXIA PerfectStorm to generate both normal and malicious traffic then Tcpdump tools are used to capture 100 GB of raw network traffic. All 49 features which are proven to be better than the older KDD99 dataset in [13] are then extracted by applying Argus, Bro-IDS tools, and twelve algorithms.

The full dataset contains more than 2.5 million records split into 4 CSV files and the total 49 features are grouped into five categories: flow, basis, content, time and additional generated features as in Table 1. Network traffic is labeled as either benign (normal) or malicious, with more than 87% of the full dataset being normal traffic records.

Table 1. Features of UNSW-NB15 Dataset

#	Feature Category	Feature Name
1	Flow Features	srcip, sport, dstip, dsport, proto
2	Basic Features	state, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, service, sload, dload, spkts, dpkts
3	Content Features	swin, dwin, stcpb, dtcpb, smeanz, dmeanz, trans_depth, res_bdy_len
4	Time Features	sjit, djit, stime, ltime, sinpkt, dinpkt, tcprtt, synack, ackdat,

		is_sm_ips_ports
5	Additional Generated Features	ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd, ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm

The attack types are also categorized into nine classes as follows: Analysis, Backdoors, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms. The distribution of each attack class is illustrated in Fig.1

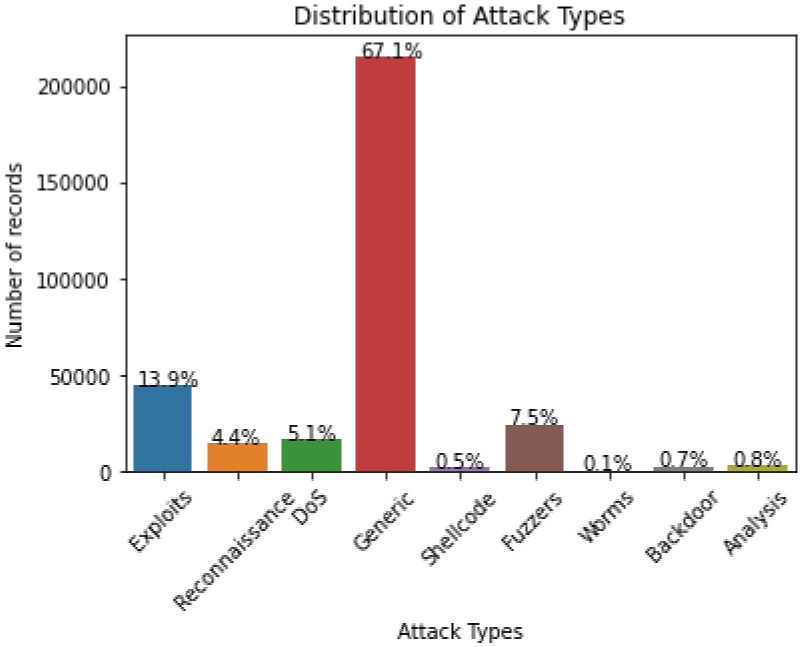

Fig. 1. Distribution of Attack Types in UNSW-NB15 Dataset

For the full dataset, data preprocessing was performed in the manner below:

- Merge four CSV files into one.
- Column labels cleaned up to trim off the extra whitespaces.
- Whitespace in string values cleaned.
- Drop duplicate records.
- Replace '-' and null attributes with 0.
- Empty 'attack_cat' values in benign records replaced with 'Normal'.

- Merges the "Backdoors" typo into the "Backdoor" attack category.
- Transform the "Label" feature from numeric data type to Nominal one.

Moreover, the partitions of the full dataset are provided which splits into a training set and a testing set. There are 175,341 records in the training set whilst 82,332 records present in the testing set. Six features are removed from the original dataset ("srcip", "sport", "dstip", "dsport", "stime" and "ltime") and two new features are added ("id" and "rate"). Unlike the full data set, there are no missing records within this data. All kinds of combined network traffic (training and testing set) are distributed as in Fig.2.

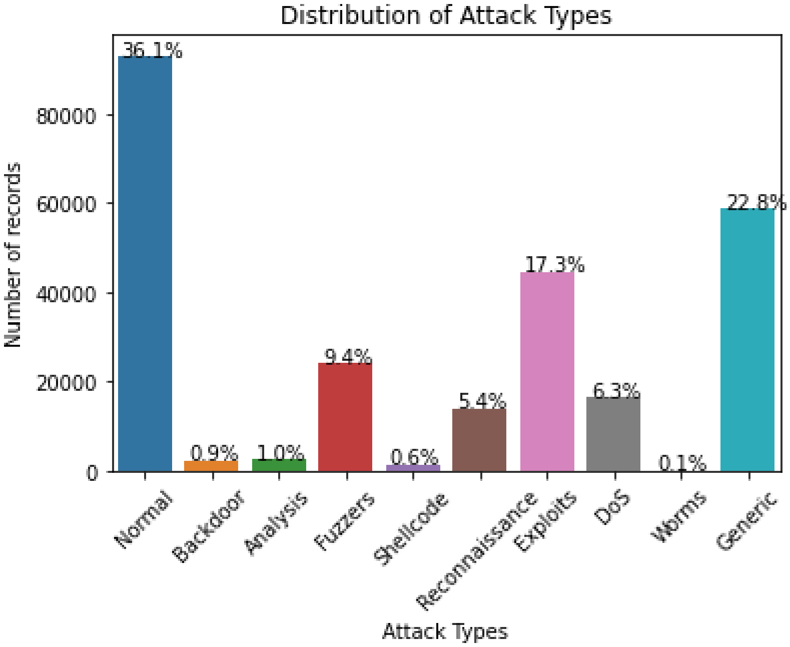

Fig. 2. Distribution of Attack Types in the Prepared Portion of the Dataset

In this project, we will utilize both the full and prepared dataset which are transformed using the preprocessing tools provided in Weka and Scikit-Learn. Notice that as the total number of features and their domains may differ in the two dataset, the steps to process the features are the same. Firstly, the *StringToNominal* unsupervised filter in Weka is applied to String type features (Stime, Ltime). Secondly, since DL algorithms work on numeric features only, a *LabelEncoder* is applied to normalize the labels (e.g: service, dstip, proto, etc) into values ranging from 0 to n_classes-1. This step is done by libraries found in Scikit-Learn. Additionally, feature scaling through standardization (Z-score normalization) is also employed within the data preprocessing procedure to center the numeric records into zero mean and unit standard deviation. This is done by employing the *Standardize* filter provided in Weka. The purpose of this step is to make sure that all the features are in the same scale or, in order words, the importance of all features are considered to be the same. Additionally, some of the distance-based ML algorithms are known to be sensitive to the range of features. The terms partitioned, prepared, and partial dataset are used interchangeably to refer to the segment of the set that was already prepared into smaller training and testing sets, while the terms full and complete dataset refer to the whole dataset containing over 2 million records. Additionally, we will use a subset of important features measured by the Gedeon method used in [3].

VI. Methodology

Machine Learning

The Weka program is used for training and evaluating some conventional machine learning methods as it comes with numerous implementations included. Prior research has used the prepared subset of the data to train and test these methods, so the full dataset of over 2 million records was used here for each of the methods. A few decision tree algorithms were used, including RandomTree, REPTree, and J48. KNN was also run on the same dataset. The nature of the Decision Tree algorithms for a large and multi-dimensional dataset meant that running them anything less than a workstation results in an out of memory exception in Weka. Due to this, a ComputeCanada node with 512 GB of RAM was used to run all the methods explored in this paper.

Table 2. Training and Testing Machine Details

Processor	Intel Xeon @ 2.1 GHz
RAM	512 GB

The J48 algorithm was the quickest to train with a time of 9 minutes and 20 seconds and used 4.74 GB of RAM. It was run with a pruning confidence of 0.25, a minimum number of instances per leaf of 2, 3 folds for reduced error pruning, and a seed of 1. RandomTree took 9 minutes and 53 seconds to run to completion and used 126 GB of RAM. It ran with a minimum number of instances per leaf of 1, a minimum variance for split of 0.001, a seed of 1, and a maximum depth of 10. The REPTree algorithm took the longest at 1 hour and 16 seconds and used 406GB of RAM. It was run with a minimum number of instances per leaf of 2, a minimum variance for split of 0.001, 3 folds for reduced error pruning, and a seed of 1. All the Decision Tree algorithms were run with a 66% training 33% testing split on the full dataset. The completed trees have sizes of 195936, 917581, and 135362 for J48, RandomTree, and REPTree respectively. The KNN algorithm does not have a specific training phase, instead, it looks at all the training instances when making a prediction on an unseen record. Therefore the 'training' time of the KNN algorithm is of little use, and the prediction time will naturally be larger than that of the other methods explored in this paper.

Deep Learning

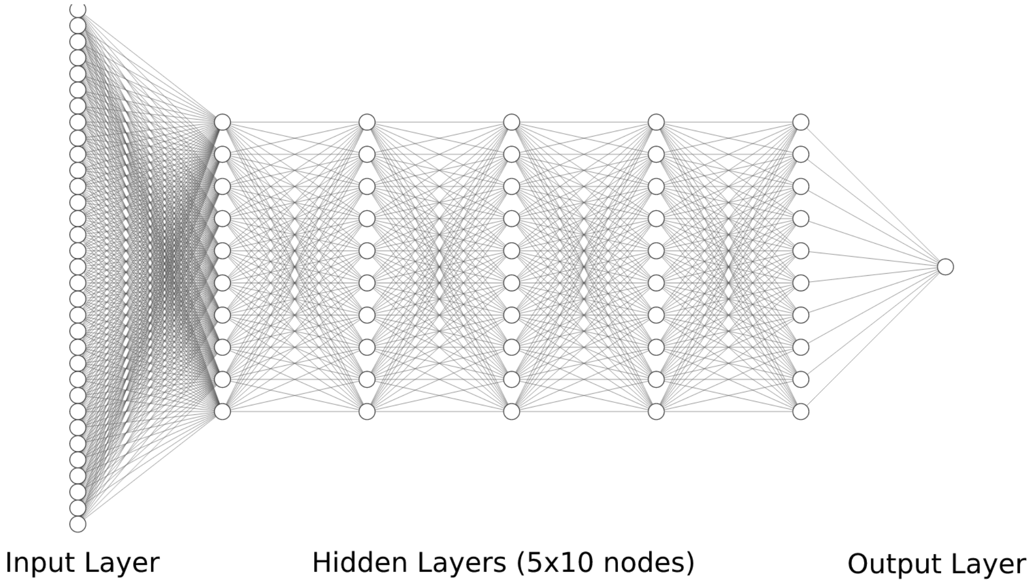

Fig. 3. Architecture of Multi-Layer Feed Forward Neural Network

A number of deep learning methods were chosen to be compared. The first was the multi-layered feed forward neural network proposed by Al-Zewairi, Almajali, and Awajan in [3]. After removing everything except the top 20% of features chosen by the Gedeon method, we are left with 33 features plus the label. The model has five hidden layers, with 10 neurons each for a total of 50 hidden neurons. The referenced paper found that the ReLU activation function had the lowest training time as well as convergent results. Therefore the ReLU function was used as the activation function for the hidden layers of the model. The Sigmoid function was used for the output layer since we are performing binary classification. The model was trained using 60%

training, 10% validation, and 30% testing split as proposed by the paper. The model was built in Google Colaboratory using TensorFlow and Keras for the architecture and training of the model, SkLearn for the stratified k-fold cross-validation, and Pandas for dataset manipulation. It was then rerun on a ComputeCanada node so that the build and prediction times could be compared to the ones measured for the conventional machine learning methods. The model was built and tested on both the partial and the complete datasets separately.

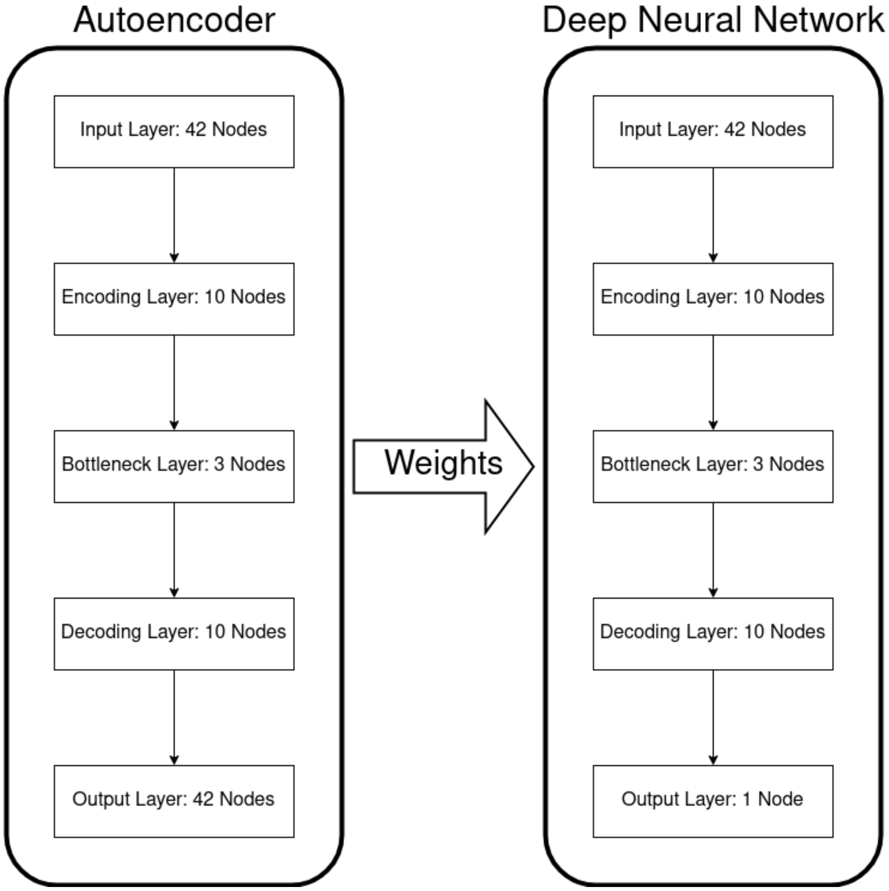

Fig. 4. Architecture of Deep Autoencoder to Deep Feed Forward Neural Network System

The second deep learning method to be implemented and compared used a deep autoencoder (DAE) to learn the structure and encodings of the records in an unsupervised learning environment proposed by Muna AL-Hawawreh, et al. in [5]. The DAE's architecture consisted of an encoding structure using an input layer of 42 nodes (corresponding to the number of features of the records) and a hidden layer of 10 nodes, the bottleneck layer had three nodes, and the decoding structure consisted of a hidden layer of 10 nodes and an output layer of 42 nodes. The DAE was trained on 20% of the dataset, consisting of only 'normal' labeled records.

The top output layer of the deep autoencoder was then removed and replaced with an output layer used for binary classification (1 node). This adjusted model was then trained on 60% of the remaining records and tested on the last 20%. This model was also built in Google Colaboratory, using Keras and Tensorflow for the architecture of the model, SkLearn to help split the training and testing data as well as some preprocessing, and Pandas for other types of dataset manipulation and exploration. As with the previously mentioned deep learning model, this model was also rebuilt and retested on a ComputeCanada node so that the build and testing times can be compared. Likewise, this algorithm was evaluated with the full data set as well. The only change being that the full dataset was repartitioned to have the same ratio of normal to anomalous records as was used in the prepared dataset.

The goal of this project is to conduct a comprehensive comparison of different approaches using both machine learning and deep learning techniques for NIDS in terms of accuracy and execution time. To the best of our knowledge, the first proposed paper is chosen as it achieves the highest accuracy for anomaly detection. We use a deep autoencoder as the second deep learning method since it might allow for learning different encodings of the features and catch some anomalous records that might be missed in a conventional feed-forward neural network. It can also be assumed that the classification time will be quicker when using the deep autoencoder since it consists of fewer layers and nodes.

Batch Prediction Time

In real-time processing NIDS, volume (amount of data) and velocity (data processing time) are the key concerns since the network size, capacity and speed within an enterprise keep increasing. Thus, one of the problems of NIDS is to be able to detect the intrusion as soon as possible. Within the local network of a company, traffic flows are often aggregated within the choke-points (ingress router or switch devices). Typically, they have limited resources so in order to avoid overheads incurring when collecting network data, random sampling techniques [14] could be applied to select a given size of data. The batch of data can then go through the process of feature extraction and prediction. Therefore, the prediction time of a batch of traffic flows has a critical impact on the NIDS performance. An example of anomaly detection for the batch traffic flows using machine learning approach is illustrated in Fig. 5. Network records are sampled at the choke-points using *tcpdump* tools, then the resulting *pcap* files are sent to

Bro-IDS or Argus for feature extraction. Finally, the IDS loaded with the train model performs the anomaly detection. Moreover, the trained model could be continuously retrained and updated.

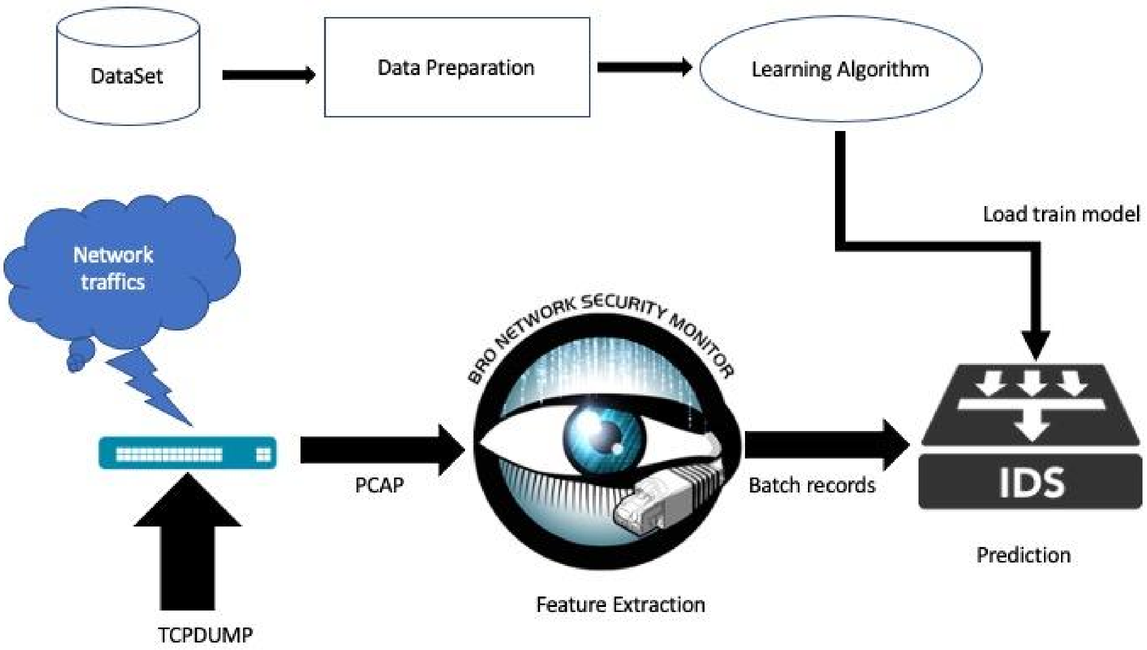

Fig. 5. Anomaly Detection Process for batch network.

VII. Results

Both the deep learning and the conventional machine learning methods were tested on unseen subsets of the UNSW-NB15 dataset, and the results organized in Tables 3 and 4. The prediction times for a single record were recorded by measuring the time it took to predict the labels of a whole set of records using each model, and dividing it over the number of records used. The training and single prediction times are shown in Table 5. The main metrics of evaluations were the accuracy, precision, recall rate, false negative rate, and the false alarm rate (otherwise known as the false positive rate).

The accuracy can be calculated as:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} \tag{1}$$

The precision can be calculated as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2}$$

The recall rate can be calculated as:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

(3)

The false negative rate can be calculated as:

$$False\ Negative\ Rate = 1 - Recall = \frac{False\ Negatives}{False\ Negatives + True\ Positives} \quad (4)$$

The false alarm rate or false positive rate can be calculated as:

$$False\ Alarm\ Rate = \frac{False\ Positives}{False\ Positives + True\ Negatives} \quad (5)$$

Since the models are trained on a subset of the data, and tested on another separate unseen subset, it can be safely assumed looking at the results of the deep learning methods in Table 3 that using the complete dataset resulted in higher accuracy while not overfitting. The multi-layered feed forward (MLFF) model trained on the complete set of records has one of the lowest False Alarm Rates of the models tested, and so is in consideration to be the best option. The Deep Autoencoder model's high recall rate means the false negative rate for that model is very low. This is important since it might be dangerous for a NIDS to miss an actual anomalous record. For this reason, and the fact that the prediction time for a single record is shorter when using the Deep Autoencoder model, we can consider the Deep Autoencoder model to be the most useful of the deep learning models we explored.

Table 3. Results of All Binary Classification Methods (Running on ComputeCanada Node)

	Accuracy	**Precision**	**Recall**	**False Positive Rate (FAR)**
Multi-Layered Feed-Forward with partial data	92.22%	92.83%	95.17%	13.01%
Multi-Layered Feed-Forward with complete data	99.06%	97.40%	95.13%	0.37%
Deep Autoencoder with partial data	92.52%	93.71%	94.28%	10.35%
Deep Autoencoder with complete data	99.46%	99.16%	99.99%	1.40%

J48	96.87%	97.23%	97.40%	0.40%
REPTree	99.28%	97.47%	96.85%	0.36%
Random Tree	99.23%	97.03%	96.85%	0.43%
KNN	99.27%	97.42%	96.82%	0.37%

The conventional machine learning methods all had very similar results when trained on the complete set of records (removing only a small subset for testing). Therefore, the prediction time for a single record plays a larger part in picking a model. The KNN model has the longest single prediction time at 0.18 sec, which is four magnitudes greater than the average prediction times of the rest of the models. The RandomTree model has the fastest prediction time at 3.51 ns, with the second best being the J48 model with a prediction time of 6.23 ns. Both the RandomTree and the J48 models' prediction times are still significantly less than the rest of the binary classification models explored, and one might pick either of the models when designing a system.

Table 4. Results of All Multiclass Classification Methods

	Average Weighted Accuracy	**Average Weighted Precision**	**Average Weighted Recall**	**FAR**	**False Negative Rate**
Multiclass Multi-Layered Feed Forward	91.1%	97.0%	96.0%	0.57%	5.57%
Multiclass J48	98.0%	98.1%	98.0%	0.24%	4.49%
Multiclass REPTree	97.9%	97.9%	97.9%	0.34%	3.58%
Multiclass RandomTree	97.7%	97.7%	97.7%	0.51%	3.37%

When comparing the Deep Autoencoder to the J48 or RandomTree models, the purpose of the system being built must be looked at in more detail. If the system is handling sensitive data, a focus on security might be in order, and the Deep Autoencoder's higher accuracy and lower false negative rate might be more important than the prediction time. On the other hand, if the system deals with a high volume of packets and connections, the conventional machine learning models' substantially lower prediction times might make a difference in the user experience.

Table 5. Training and Testing Time of Models

	Training Time	**Prediction Time**
Multi-Layered Feed-Forward with partial data	1.27 mins	14.68 ns
Multi-Layered Feed-Forward with complete data	11.66 mins	14.56 ns
Deep Autoencoder with partial data	6.68 mins	17.05 ns
Deep Autoencoder with complete data	24.01 mins	11.86 ns
J48	9.34 mins	6.23 ns
REPTree	60.27 mins	10.92 ns
Random Tree	9.89 mins	3.51 ns
KNN	<1 mins	0.18 sec

Another option that was tested, was to train the decision tree algorithms to predict whether the record is normal or anomalous, but to also predict the type of anomaly. This option means the accuracies of the models are decreased compared to their binary classification counterparts, and more importantly, the false negative rate is found to be higher. Using the confusion matrix in Table 6, it can be seen that some of the models fail to correctly predict the majority of the records of some of the anomaly types.

Table 6. Confusion Matrix for Multiclass J48

Classified as	

a	b	c	d	e	f	g	h	i	j	
750298	1038	51	51	22	11	2592	0	2	90	**a=Normal**
324	12148	419	1698	247	5	276	26	21	18	**b=Exploits**
47	911	3552	170	12	3	54	4	3	2	**c=Reconnaissance**
62	4172	59	1203	71	1	82	3	8	5	**d=DoS**
47	791	25	206	72208	2	51	4	0	4	**e=Generic**
13	465	13	0	3	13	2	0	0	0	**f=Shellcode**
2676	832	51	159	26	2	4508	1	8	1	**g=Fuzzers**
0	38	2	1	4	0	0	17	0	0	**h=Worms**
14	531	2	114	5	0	61	0	55	0	**i=Backdoor**
92	534	0	127	7	0	61	0	0	79	**j=Analysis**

The multi-layered feed forward neural network was also trained for the same purpose by replacing the 1 node output layer with one that had 10 nodes, one for each traffic type. The model was retrained and compared against the multiclass classification conventional machine learning methods mentioned earlier for both FAR and false negative rate as well as the training and prediction times. As shown in Table 7, this model was the lowest scoring in all metrics when compared to the other multiclass classification methods, with the J48 decision tree algorithm again coming out as the probable best solution to the problem.

Table 7. Multi-Layered Feed-Forward with complete data

Classified as										
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	
703	0	13	67	0	0	20	0	0	0	**a=Analysis**
660	0	12	22	0	0	5	0	0	0	**b=Backdoor**

4102	0	92	651	4	3	54	0	0	0	**c=DoS**
6947	0	135	6128	0	0	139	0	0	0	**d=Exploits**
5620	0	17	35	50	9	1543	0	0	0	**e=Fuzzers**
1129	0	14	554	0	62847	100	0	0	0	**f=Generic**
6188	0	1	544	74	10	658812	0	0	0	**g=Normal**
3726	0	19	376	0	35	40	0	0	0	**h=Reconnaissance**
440	0	0	0	0	0	13	0	0	0	**i=Shellcode**
22	0	0	31	0	0	0	0	0	0	**j=Worms**

VIII. Conclusion

In this project, two deep learning models, one of which is a multi-layer feed forward neural network and the other a combination of a deep autoencoder and a deep feed forward network, have been evaluated and compared with supervised machine learning techniques for anomaly detection and malicious activity classification. We not only focus on the accuracy and FAR but also on the prediction time for each record. When performing binary classification to predict whether a record is normal or anomalous, the deep autoencoder trained on the full dataset and the J48 decision tree methods both have very high accuracies. The autoencoder also has a very high recall rate (very low false negative rate) but it's FAR might be too high depending on the use case of the system. The training times vary from model to model, but more importantly, the prediction time of a single record when using the J48 decision tree takes almost half as long as predicting a single record using the deep autoencoder. Therefore, when performing binary classification between normal and anomalous traffic records, using either the J48 decision tree algorithm or the deep autoencoder model comes down to a matter of deciding what is more important in the system being built. A short exploration into the multiclass classification problem was also done, with no clear advantage to any one conventional machine learning method.

Validity of Findings

When parsing the results of the methods that were tested, it is important to take into account the fact that the unbalanced UNSW-NB15 dataset used for training and testing these models has a high ratio of normal to anomalous records (around 6.6) and although this may or may not reflect real-world conditions, it does have an effect on the FAR and the recall. Looking at the results and confusion matrices of the multiclass classification methods show that some of the attack types have very low precision and recall. This is probably due to the low amount of records of those attack types which hinders the training of the models.

Future Work

A number of things are still to be explored regarding the prediction times and the methods used. When trying to predict the attack type of the anomalous record, one cascade approach would be to predict whether the record is normal or malicious first (binary classification) and then predict the attack type of the anomalous record after that (9-way multiclass classification). This might result in a lower false alarm rate and false negative rate, but a significantly longer prediction time might be the tradeoff. The development of a dataset that includes a larger and more balanced distribution of attack type records would also help in training and testing a multiclass model.

References

1. Rob Sobers, "110 Must-Know Cybersecurity Statistics for 2020, " Oct. 26, 2020. [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/ [Accessed on Dec. 14, 2020].
2. Warwick Ashford, "UK cyber crime growing exponentially," Apr. 12, 2016. [Online]. Available: https://www.computerweekly.com/news/450281086/UK-cyber-crime-growing-exponentially [Accessed on Dec. 14, 2020]
3. Das A., Ajila S.A., Lung CH. (2020) A Comprehensive Analysis of Accuracies of Machine Learning Algorithms for Network Intrusion Detection. In: Boumerdassi S., Renault É., Mühlethaler P. (eds) Machine Learning for Networking. MLN 2019. Lecture Notes in Computer Science, vol 12081. Springer, Cham, doi: 10.1007/978-3-030-45778-5_4.
4. Murphy KP. Machine learning: a probabilistic perspective. Cambridge (MA): MIT Press; 2012.
5. Retrieved from https://www.cs.waikato.ac.nz/ml/weka/ [Accessed on Dec 14 2020].
6. S. Kalmegh, "Analysis of WEKA data mining algorithm REPTree, simple CART and RandomTree for classification of Indian news," *International Journal of Innovative Science, Engineering, and Technology*, vol. 2, no. 2, pp. 438–446, 2015.

7. A. Zhang, Z.C. Lipton, M. Li, A.J. Smola, Dive into deep learning, Unpublished Draft. Retrieved 19 (2019) 2019.
8. Nawir M, Amir A, Lynn OB, Yaakob N, Badlishah Ahmad R. Performances of machine learning algorithms for binary classification of network anomaly detection system. J Phys: Conf Ser. 2018;1018:012015. https://doi.org/10.1088/1742-6596/1018/1/012015.
9. Alin, F., Chemchem, A., Nolot, F., Flauzac, O., Krajecki, M.: Towards a hierarchical deep learning approach for intrusion detection. In: S. Boumerdassi, E. Renault, P. M¨uhlethaler ´ (eds.) Machine Learning for Networking, pp. 15–27. Springer International Publishing, Cham (2020).
10. Muna, A. H., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. Journal of Information Security and Applications, 41, 1-11.
11. M. Al-Zewairi, S. Almajali and A. Awajan, "Experimental Evaluation of a Multi-layer Feed-Forward Artificial Neural Network Classifier for Network Intrusion Detection System," *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, Amman, 2017, pp. 167-172, doi: 10.1109/ICTCS.2017.29.
12. N. Moustafa, J. Slay, UNSW-Nb15: a comprehensive data set for network intrusion detection systems (UNSW-nb15 network data set), in: Military Communications and Information Systems Conference (MilCIS), 2015, IEEE, 2015, pp. 1–6.
13. N. Moustafa and J. Slay, "The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems," *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, Kyoto, 2015, pp. 25-31, doi: 10.1109/BADGERS.2015.014.
14. N. Moustafa, J. Hu, and J. Slay, “A holistic review of network anomaly detection systems: A comprehensive survey,” J. Netw. Comput. Appl., vol. 128, pp. 33–55, Feb. 2019.