# Cloud-based Real-time Network Intrusion Detection Using Deep Learning

Santhosh Parampottupadam, and Arghir-Nicolae Moldovann
School of Computing, National College of Ireland, Mayor Street, IFSC, Dublin 1, Ireland
E-mail: santhosh.parampottupadam@student.ncirl.ie; arghir.moldovan@ncirl.ie

***Abstract*—Deep learning has increased in popularity with researchers and developers investigating and using it for various use cases and applications. This research work focuses on real-time network intrusion detection by making use of deep learning. A cloud-based prototype system was developed to investigate the capability of deep learning based binomial classification and multinomial models to detect network intrusions in real-time. An evaluation study was carried out using the benchmark NSL-KDD dataset to compare deep learning models built using H2O and DeepLearning4J libraries, with other commonly used machine learning models such as Support Vector Machines, Random Forest, Logistic Regression and Naïve Bayes. The results showed that the choice of the deep learning library is an important factor to consider for real-time applications. The H2O deep learning based binomial and multinomial models generally outperformed the other models, achieving over 99.5% accuracy using cross-validation on the NSL-KDD training dataset and over 83% accuracy on the test dataset.**

***Index Terms*—Network security, intrusion detection, deep learning, cloud computing, NSL-KDD.**

## I. Introduction

Internet and related technologies such as networking and cloud computing are the backbone of a multitude of applications and services that are used for every aspect of modern life, from work, education to entertainment. Ericsson [1] estimated that by 2020 there will be over 8.6 billion connected phones, 1.7 billion connected PCs and tablets, and 18.1 billion IoT devices. With the increasing dependence on online services, cyber security attacks represent a major risk facing users and businesses [2]. Notorious cyber attacks include the Yahoo breach when 3 billion user accounts were compromised, and the Target stores breach when the credit card information of 110 million users was compromised [3]. While organizations are working hard build better security features by incorporating recent attack types, new attacks are continuously emerging. In this context there is increasing need for new solutions to predict and prevent network intrusion attacks in real-time.

According to Wang and Jones [4], intrusion detection can be categorized into 3 types based on its detection mechanism: anomaly-based detection, signature based detection, and hybrid detection which is a combination of the other two. These detection systems are based on studying the network parameters and body contents once an attack has happened. The anomaly based detection systems analyse the attacks payload information, and other information such as the source and destination port and internet protocol addresses to build the anomaly model [5]. These models tend to be efficient when the same attack comes again into the system, as these models are built based on the previous attack parameters. But as new attacks arise more effective self-learning models should be implemented in order to enable systems to predict and prevent yet unseen attacks.

Deep learning has emerged as a new solution that has the potential to provide more effective network intrusion detection as it uses algorithms such as feed forward and back propagation [6]. Smith [7] described best practices for building novel applications using machine learning models and suggested developing deep learning based neural networks for intrusion detection.

This paper investigates the capability of using deep learning models for network intrusion detection in real-time. A cloud hosted prototype system was developed that combines a deep learning binomial classification model to predict if there is an intrusion, with a multinomial model to identify the attack category. The prototype system integrates deep learning models built using the H2O framework [8], as well a messaging service to alert the network administrator. An evaluation study was carried out using the well-known benchmarked NSL-KDD dataset [9] to compare the H2O deep learning models with models built using DeepLearning4J, LibSVM, Random Forest, Logistic Regression and Naïve Bayes. The results showed that H2O deep learning models generally outperformed the other models, achieving over 99.5% accuracy using cross-validation on the training dataset and over 83% accuracy on the test dataset, for both binomial and multinomial classification.

The rest of the paper is structured as follows. Section II presents previous research works on network intrusion detection. Section III present the methodology of this research study, while section IV describes the prototype system. Section V presents the evaluation results, while section VI concludes the paper and presents future work directions.

## II. Related Work

### A. Anomaly Based Network Intrusion Detection

Anomaly based intrusion detection is usually the first stage of defence [4]. Cui and He [10] proposed a solution that used Hadoop MapReduce to process the input data and supply it to the Weka machine learning framework. The drawbacks are that the entire MapReduce iteration must be run again when

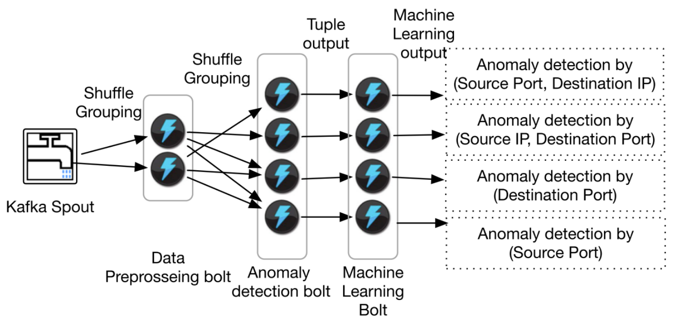

Kafka Spout
Shuffle Grouping
Data Preprosseing bolt
Shuffle Grouping
Anomaly detection bolt
Tuple output
Machine Learning Bolt
Machine Learning output
Anomaly detection by (Source Port, Destination IP)
Anomaly detection by (Source IP, Destination Port)
Anomaly detection by (Destination Port)
Anomaly detection by (Source Port)

Fig. 1. Real-time network anomaly detection system using Storm, Kafka and Weka [11].

the test results are not satisfied, and the solution not being suitable for real-time intrusion detection since MapReduce is using batch processing.

Zhao et al. [11] has addressed the batch processing delay by converting this to real-time streaming as shown in Fig. 1. The streaming data is processed using Apache Kafka and the output stream is passed to Hadoop MapReduce which uses the Weka SVM library to classify the data. The machine learning algorithm produces 4 outputs based on source port vs. destination IP, source IP vs. destination port, destination port, and source port. The authors achieved 91% accuracy.

Timcenko and Gajin [12] analysed the performance of different ensemble classifiers for supervised anomaly based intrusion detection. The compared algorithms were: Bagged trees, AdaBoost, RUSBoost, LogitBoost and GentleBoost. The authors applied random under-sampling boosting to deal with class imbalance, and concluded that Bagged tree and GentleBoost achieved the highest classification performance from the compared algorithms.

### B. Signature Based Network Intrusion Detection

Along with the anomaly based detection, signature based detection is also used in many security defence systems. These systems analyse the signatures of the request based on factors such as port, source and destination IP address, payload information of the request along with metadata.

Modi et al. [5] indicated that signature based intrusion detection solutions are efficient and able to detect attacks when the stored or predefined attacks come again. Each time these solutions will check with the stored signature based model to identify the anomalies. Manohar Naik and Geethanjali [13] proposed a multi-fusion pattern matching algorithm that combines the Berry-Ravindran and Raita algorithms. While the proposed algorithm is more efficient, it is not real-time as there was a time delay to detect the signature based attack types. Jongsuebsuk et al. [14] addressed the time delay by using a Fuzzy Genetic Algorithm and using only 12 parameters from the KDDCup1999 dataset for training the algorithm. The authors achieved 97.5% accuracy and ability to detect the intrusion with a 3 seconds delay.

Rathore [15] has also addressed real-time intrusion detection and used Hadoop to built a machine learning model using SVM, Naïve Bayes, and Conjunctive rule. MapReduce was implemented to iterate over the network data and filter the unwanted data using Conjunctive rule. The final output of MapReduce is sent for validation to the decision system which classifies it into intrusion or not intrusion. The authors claim that the SVM took 120 seconds to build the model. One important aspect to note is that the authors have only used 6 parameters from KDDcup99 dataset to develop the model.

### C. Deep Learning Based Intrusion Detection

A number of researchers have explored deep learning for intrusion detection. Kim et al. [16] used deep neural network (DNN), and basic data cleaning and duplicate elimination features to pre-process the data along with training data conversion. Since the KDDcup99 dataset contains integers and floating values, they converted all the data into string type to minimize the data loss. The authors used 10% of the corrected data for training the model and achieved 99% accuracy with 0.08% false alarm rate.

Saxe and Berlin [17] used more than 400k software binaries for building a deep learning model consisting of four layers. The first layer is training layer, the second is categorizing the data based on labelled data and unlabelled data, while the final layer is for classification. The researchers achieved 95% accuracy, but data cleansing was not mentioned, and this is important given that the KDDcup99 dataset contains more than 70% duplicate records [9].

LeCun et al. [18] used a recurrent neural network which takes the input in sequential mode. When each row passes through the system it holds the vector state of the previous result and train with the previous model. The author mentions that the training process is difficult since the vector gradient values might compress or break during the training phase.

Li et al. [19] used an auto encoder based approach and the same benchmarked dataset KDDcup99 to train and test the model. The auto encoder was used along with the deep belief network and normalized training data to reduce the log loss. Auto encoder based approach and normalized approach gave a better result than the recurrent neural network approach. While both methods provided good results they are not real-time.

Salama et al. [6] implemented a Restricted Boltzmann Machine System. It was working based on combining the already existing SVM algorithm and deep belief network. The input dataset flows through the system which extracts 13 network parameters that are used as input by the SVM to classify the input into intrusion or not intrusion.

Niyaz et al. [20] proposed a self taught learning mechanism which has three stages. The feature learning is done from the first layer which is using unlabelled data. The learned vectors are applied on labelled dataset in the second stage, followed by a softmax regression and classification to predict the network intrusion detection. The authors state that they used softmax regression since it has high efficiency and is easily implementable compared to other regression models. The work was done on Matlab based system, with three levels of data cleaning process to reduce the log loss. They obtained

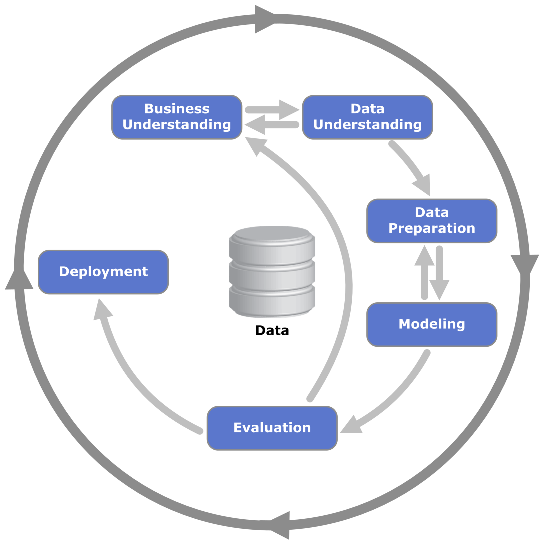


Fig. 2. Cross-industry standard process for data mining (CRISP-DM) [22].

an F-Measure of 98.84% on test data and 88.39% on training and test data. The authors concluded the research by stating that future work would be to convert the model into real-time intrusion detection system.

Aygun and Yavuz [21] proposed two machine learning models for anomaly detection, namely auto encoder and denoising auto encoder (DAE) respectively. The main difference between the two models is in the way how they perform on inputs. The DAE is randomly corrupting its input data and it will try to regenerate it using stochastic methods, hence it will try to learn its statistical relationship with the input dataset. The authors used the same KDDcup99 dataset for evaluation and achieved 88.86% accuracy.

The literature review showed that the proposed intrusion detection solutions offer different performance level, and deep learning solutions may not necessarily outperform other traditional machine learning algorithms. Therefore, more research is needed to investigate the capabilities of deep learning. Moreover, many of the proposed solutions are not real-time. As such, this research will focus on investigating the capabilities of deep learning for real-time intrusion detection as compared to other machine learning models.

## III. Methodology

The research methodology followed the cross-industry standard process for data mining (CRISP-DM), which is a sequential approach from business requirement collection to deployment and consists of 6 phases as shown in Fig. 2 [22].

### A. Problem Understanding

The literature review was carried out in order to understand the research problem as well as existing solutions that were proposed previously. The literature review showed that the capabilities of deep learning for real-time intrusion detection have not been thoroughly investigated in the literature and more research is needed. As such, this research focuses on real-time intrusion detection and proposes to combine a deep learning binomial classification model to predict if there is an intrusion, with a deep learning multinomial model to identify the attack category.

TABLE I
Table 3: NSL-KDD Intrusion Attack Type Classification.

| Class | Attack Type |
|---|---|
| DoS | apache2*, back, land, mailbomb*, neptune, pod, processtable*, smurf, teardrop, udpstorm*, worm* |
| Probe | ipsweep, mscan*, nmap, portsweep, saint*, satan |
| R2L | ftp_write, guess_passwd, httptunnel*, imap, multihop, named*, phf, sendmail*, snmpgetattack*, snmpguess*, spy$^\dagger$, warezclient$^\dagger$, warezmaster, xlock*, xsnoop* |
| U2R | buffer_overflow, loadmodule, perl, ps*, rootkit, sqlattack*, xterm* |

Note: $^\dagger$ indicates attack types present only in training data
* indicates attack types present only in test data

### B. Data Understanding

The literature review showed that most related works on intrusion detection have used the benchmark KDDcup99 dataset [23]. However the drawback of using this dataset is that it has repeated values which might affect the models' training. According to Tavallaee et al. [9] the KDDcup99 train and test datasets have 78% and 75% duplicated records respectively, thus even basic machine learning models can achieve over 98% accuracy on train data and 86% accuracy on test data.

Therefore this research uses the NSL-KDD dataset that consists of selected records of the KDDcup99 dataset and does not suffer of the same shortcomings [9], [24]. The NSL-KDD train dataset has 41 attributes, 3 nominal (i.e., protocol, service, flag), and 38 numerical (e.g., duration, source bytes, destination bytes, wrong fragment, number failed logins, etc.). The dataset consists of normal traffic and 39 attack types that are categorized into four groups, namely: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Remote (U2R) [25]. As shown in Table I, 20 attack types are present in both train and test datasets, 2 attack types are only present in the train dataset, while 17 attack types are only present in test dataset. The reason was to enable evaluating the capabilities of algorithms to detect new attack types assuming that these are often variants of known attacks [9].

Table II presents the distribution of records for normal traffic and the four attack classes. Moreover, Table III presents the distribution of records for the top 10 attack types from the train and test datasets.

### C. Data Preparation

The NSL-KDD dataset comes in a structured format and is a clean sample of the KDDcup99 dataset, thus it requires little preprocessing. The train and test datasets were checked for missing values but all records were complete, thus no record elimination or imputation was necessary. Some machine learning algorithms only work on numeric data, so the nominal attributes have to be mapped into multiple binary numeric

TABLE II
DISTRIBUTION OF RECORDS FOR NORMAL TRAFFIC AND DIFFERENT ATTACK CLASSES.

| | **Train Data** | | **Test Data** | |
|---|---|---|---|---|
| **Class** | **Records**[#] | **Records** [%] | **Records**[#] | **Records** [%] |
| Normal | 67343 | 53.46 | 9710 | 43.07 |
| Intrusion | 58630 | 46.54 | 12833 | 56.93 |
| DoS | 45927 | 36.46 | 7460 | 33.09 |
| Probe | 11656 | 9.25 | 2421 | 10.74 |
| R2L | 995 | 0.79 | 2885 | 12.80 |
| U2R | 52 | 0.04 | 67 | 0.30 |
| Total | 125973 | 100.00 | 22543 | 100.00 |

TABLE III
DISTRIBUTION OF RECORDS FOR NORMAL TRAFFIC AND DIFFERENT ATTACK TYPES.

| | **Train Data** | | **Test Data** | |
|---|---|---|---|---|
| | **Traffic** | **Records** [%] | **Traffic** | **Records** [%] |
| | normal | 53.46 | normal | 43.07 |
| Top 10 Attack Types | neptune | 32.72 | neptune | 20.66 |
| | satan | 2.88 | guess_passwd | 5.46 |
| | ipsweep | 2.86 | mscan | 4.42 |
| | portsweep | 2.33 | warezmaster | 4.19 |
| | smurf | 2.10 | apache2 | 3.27 |
| | nmap | 1.19 | satan | 3.26 |
| | back | 0.76 | processtable | 3.04 |
| | teardrop | 0.71 | smurf | 2.95 |
| | warezclient | 0.71 | back | 1.59 |
| | pod | 0.16 | snmpguess | 1.47 |
| | other attacks | 0.12 | other attacks | 6.62 |

attributes using one hot encoding. However, this was done automatically by the H20 algorithms implementation.

One attribute was removed as it was 0 for all train and test records. The original datasets only specified the attack type as per Table II. Therefore, two new columns were added, one specifying if the record is normal or intrusion, the other specifying the attack class.

### D. Modeling

This research work was inspired by previous works, including that of Niyaz et al. [20] who proposed a deep learning approach for network intrusion detection and indicated that future work would be to expand the model for real-time intrusion detection. This paper is focused on deep learning based real-time network intrusion detection using cloud technologies. Arora [26] claim that even though we have multiple machine learning models, deep learning based neural network models are best because of their capability to learn new features and provide high precision and accuracy.

The proposed cloud-based network intrusion detection system combines two deep learning models:

- Deep learning based binomial classification model that is used in order to predict the intrusion from the normal network traffic.
- Deep learning based multinomial model that is used when the first model detects an intrusion to detect into which class it falls (i.e, DoS, Probe, R2L and U2R).

Two deep learning libraries were considered for comparison purposes, namely H2O Deep Learning [8], and DeepLearning4J [27]. Moreover, the performance of the deep learning models was compared to that of other machine learning algorithms integrated by Weka [28], namely: Support Vector Machines (LibSVM), Random Forest, Logistic Regression, and Naïve Bayes. All these libraries work in the Java VM, which makes for a more direct comparison.

### E. Evaluation

The evaluation of the models was done using two approaches. The first approach was using 5-fold cross validation on the NSL-KDD train data. The second approach trained the models on the train dataset without validation split, and tested the models on the test dataset. The following metrics were considered to evaluate the performance of the machine learning models:

- Accuracy: defined as the percentage of correctly classified records over the total number of records;
- Precision: defined as the ratio of true positive records divided to true positive plus false positive classified records;
- Recall: defined as the ratio of true positive records divided to true positive plus false negative classified records;
- F-measure: defined as the harmonic mean of precision and recall;
- AUC: the area under the ROC curve which is commonly used to assess the robustness of machine learning models.
- Detection rate: defined as the ratio of instances correctly classified as belonging to that particular class.

### F. AWS Deployment

A web application was built which could be used by the network/security administrator to monitor the live status of the network. The application also sends real-time mobile notifications to the admin in case of an intrusion. The application was hosted in Amazon Web Services and interacts with the H2O machine learning models via API calls. More details on the proposed system prototype are provided in the next section.

## IV. PROTOTYPE SYSTEM

### A. Architecture

Fig. 3 illustrates the prototype system that was implemented in order to showcase the real-time network intrusion detection capability of deep learning using cloud computing. A web application was developed for network monitoring, which integrates two deep learning models: binomial classification model to identify if there is an intrusion or not, and multinomial model to detect the attack class in case of an intrusion.

The application is hosted in AWS and communicates with the deep learning POJO models generated by H2O using API calls. The H20 environment was configured in AWS EC2

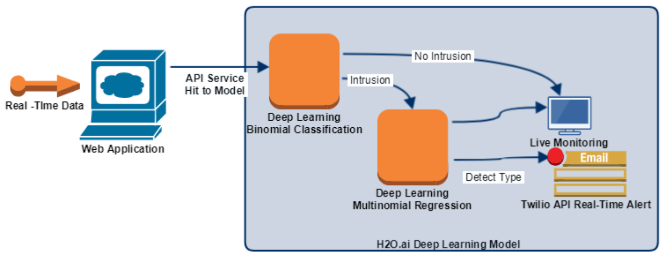


Fig. 3. Proposed real-time network intrusion monitoring and alert system using Deep Learning.

TABLE IV
SOFTWARE TOOLS USED FOR DEVELOPING THE PROTOTYPE SYSTEM WEB APPLICATION AND THE INTEGRATED DEEP LEARNING MODELS.

| Web Application Tools | | Machine Learning Tools | |
|---|---|---|---|
| IDE | Eclipse Neon | IDE | Rstudio 1.0.143 |
| Language | Java 1.8 | Language | R 3.4.2 |
| Scripting | jQuery 2.1 | ML Framework | H2O 3.16.0.2 |
| Service | J2EE | Algorithm | Deep Learning |
| Container | Jetty WebServer 9.4.7 | Dataset | NSL-KDD |
| Packaging | Gradle 4.3.1 | | |
| Front-end | Bootstrap 4 | | |
| Notification | Twilio API | | |

instance and the deep learning models were deployed in the cloud environment.

If an intrusion is detected, the application shows live alerts in the web interface. Moreover, Twilio API is integrated for sending real-time notifications to the mobile phone of the network administrator, thus enabling the admins to take quick actions even if they are not with the monitoring system.

### *B. H2O Deep Learning*

The deep learning based models were developed using the open source H2O library [8]. R language was used for developing the deep learning models. The main reason for choosing H20 is its wide acceptance for machine learning and the fact that it provides the POJO based API services to the web application. Since the web application model would be useful for network admin, the web based model is an essential part of the system.

Once the deep learning model is created the Java POJO classes can be downloaded. There are two ways to download the model, using a local host Web UI or using the H2O library. It also provides the H2O generated model jar which contains all the dependency jars to support the Java API's. In this case, one Java class for the binomial classification model and one for multinomial model are downloaded along with the H2O Generated Model jar.

### *C. Hyper Parameter Tuning*

Hyper parameter tuning is a important aspect in developing a deep learning model. Before final deployment, the deep learning models can be further optimised by testing various combination of parameter values. The activation function used for the models is Rectifier as it provides efficient computations and sparse activation. Other parameters that can be changed include the number of epochs and hidden layers. H2O also provides the AutoML library which can be used for automatic training and tuning of the deep learning models.

### *D. H2O POJO Interaction*

The Java web application needs to implement the generated deep learning model class by creating a predictor servlet. The H20 framework provides API's to interact with the Custom Java Classes. The servlet class receives the real-time data and passes it through the machine learning models. The input data is converted to H2O data format and passed to the binomial model. The deep learning binomial model predicts if it is an intrusion or not. If an intrusion is detected, it calls the multinomial model to predict the attack class (i.e, DoS, Probe, R2L and U2R). As the deep learning models are pre-trained it takes micro seconds to predict if a traffic record is an intrusion. If an intrusion is detected the application sends a mobile alert to the registered devices using Twilio API.

### *E. Development Tools*

Table IV provides a summary of the software tools used for developing the prototype system. J2EE, jQuery and Bootstrap were used for building the back-end and front-end of the web application. Eclipse IDE was used for handling the Java part of the application. R and RStudio were used for building the H2O deep learning models.

AWS EC2 was used for deploying the web application and the Java POJO deep learning models generated by H2O. Jetty server was used for deploying and testing the Java POJO models in local mode before deployment to the cloud. Twilio API and its messaging services are used for sending real-time alert to the network admin.

Gradle was used as the build tool to integrate and test the deep learning POJO models and Java resources. The newest version of the dependency integrity offered by Graddle was used. Moreover, the Shadow plugin was used for combining dependency classes and resources. The goal of a bundled library is to create a pre-packaged dependency for other libraries or applications to utilize.

## V. EVALUATION RESULTS

This section presents the evaluation results for the binomial and multinomial classification models on NSL-KDD dataset.

### *A. Binomial Models Comparison*

Tables V and VI present the performance evaluation results of the binomial machine learning models for the two cases considered: (i) using 5-fold cross validation on the NSL-KDD train dataset, and (ii) training the models on the train dataset without validation split plus testing them on the test dataset. All the models were built using the default settings of the libraries. The LibSVM was too slow for cross-validation so it was excluded from the first comparison approach.

For the cross-validation approach, all binomial models offer over 90% accuracy with the top models being Deep Learning

TABLE V
PERFORMANCE OF BINOMIAL MACHINE LEARNING MODELS ON NSL-KDD TRAIN DATASET USING 5-FOLD CROSS-VALIDATION.

| | | | | | | Detection Rate [%] | |
|---|---|---|---|---|---|---|---|
| **Model** | **Accuracy [%]** | **Precision** | **Recall** | **F-measure** | **AUC** | **Normal** | **Intrusion** |
| Deep Learning H2O | 99.52 | 99.95 | 100.00 | 99.55 | 99.94 | 99.66 | 99.43 |
| Deep Learning 4J | 96.79 | 96.80 | 96.80 | 96.80 | 99.00 | 97.05 | 96.48 |
| Random Forest | 99.91 | 99.90 | 99.90 | 99.90 | 100.00 | 99.96 | 99.86 |
| Logistic Regression | 97.08 | 97.10 | 97.10 | 97.10 | 99.40 | 98.22 | 95.77 |
| Naïve Bayes | 90.34 | 90.40 | 90.40 | 90.30 | 96.60 | 93.59 | 86.65 |

TABLE VI
PERFORMANCE OF BINOMIAL MACHINE LEARNING MODELS ON THE NSL-KDD TEST DATASET.

| | | | | | | Detection Rate [%] | | Runtime [sec] | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Accuracy [%]** | **Precision** | **Recall** | **F-measure** | **AUC** | **Normal** | **Intrusion** | **Training** | **Test** |
| Deep Learning H2O | 83.87 | 79.47 | 100.00 | 81.83 | 87.42 | 87.08 | 84.20 | 162.97 | 1.20 |
| Deep Learning 4J | 76.12 | 80.20 | 76.10 | 76.00 | 75.10 | 91.30 | 64.64 | 808.06 | 3.56 |
| LibSVM | 72.36 | 81.60 | 72.40 | 71.50 | 75.50 | 97.80 | 53.10 | 8041.31 | 242.41 |
| Random Forest | 80.25 | 85.10 | 80.30 | 80.10 | 95.30 | 97.27 | 67.37 | 111.61 | 0.87 |
| Logistic Regression | 74.58 | 79.80 | 74.60 | 74.30 | 91.40 | 92.55 | 60.98 | 202.60 | 0.45 |
| Naïve Bayes | 76.11 | 80.90 | 76.10 | 75.90 | 90.60 | 93.14 | 63.23 | 1.23 | 0.79 |

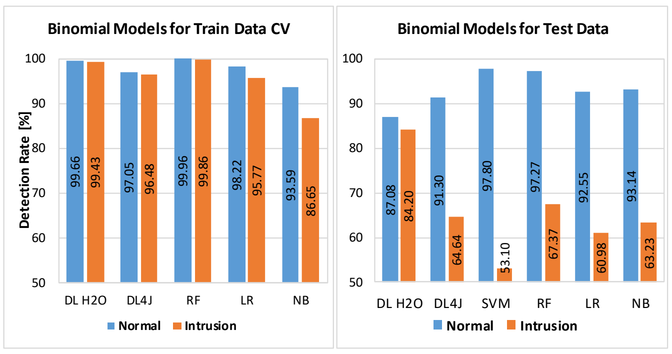


Fig. 4. Detection rates of binomial models for normal traffic and intrusion.

H20 and Random Forest at over 99.5% accuracy, while the weakest being Naive Bayes at 90.34% accuracy. In terms of detection rates again Deep Learning H20 and Random Forest are again the best models being providing over 99.4% correct detection for both normal traffic and intrusions.

For the training plus test approach, Table VI shows that all machine learning algorithms score lower on the test dataset. This is because the test data comes from a different distribution and has additional attack types not present in the train dataset (see Table III). On the test dataset, the Deep Learning H2O model provides the highest accuracy at close to 84%, followed by the Random Forest model at slightly above 80%.

Fig. 4 presents the normal and intrusion detection rates of the binomial models for the two evaluation approaches. The results show that Deep Learning H2O also provides most balanced performance with 87% and 84% detection rates for normal and intrusions respectively. While all other models provide over 90% detection rate for normal traffic, they could only detect up to 67% of intrusions. The conclusion that can be drawn is that without further optimisation, Deep Learning H2O would be the best binomial model to be implemented in the prototype system.

Table VI also shows the runtime for training and testing the models on the NSL-KDD dataset. The measurements were taken on a Windows 10 laptop with an Intel Core i5 CPU, using the default settings of the algorithm implementations. The results show that Naïve Bayes is the fastest while SVM is the slowest model to train and test. Another interesting finding is that Deep Learning H2O compares favourably to DeepLearning4J, being approximately 5 times faster to train the model. However, it should be noted that runtime was not the main focus of the evaluation, and more comprehensive testing would be required for stronger conclusions with regard to their computation efficiency and resource utilisation.

## *B. Multinomial Models Comparison*

Tables VII and VIII present the performance evaluation results of the multinomial machine learning models on the NSL-KDD train and test dataset respectively. For this evaluation, the normal records were filtered out and the models were trained on the intrusion data only, with the goal to predict the class (i.e., DoS, Probe, R2L and U2R). The multinomial models were also built with the default settings of the libraries, and without performing class balancing.

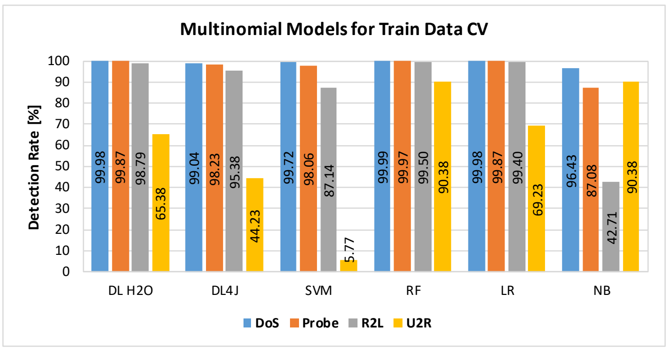
For the cross-validation approach, all binomial models achieved close to or above 99% accuracy, with the exception of Naïve Bayes who scored 93.7% accuracy. The distinction between the different multinomial models is more clear for the train plus test approach, with the top models being Logistic

TABLE VII
PERFORMANCE OF MULTINOMIAL MACHINE LEARNING MODELS ON NSL-KDD TRAIN DATASET USING 5-FOLD CROSS-VALIDATION.

| | | Detection Rate [%] | | | |
|---|---|---|---|---|---|
| **Model** | **Accuracy [%]** | **DoS** | **Probe** | **R2L** | **U2R** |
| Deep Learning H2O | 99.91 | 99.98 | 99.87 | 98.79 | 65.38 |
| Deep Learning 4J | 98.77 | 99.04 | 98.23 | 95.38 | 44.23 |
| LibSVM | 99.09 | 99.72 | 98.06 | 87.14 | 5.77 |
| Random Forest | 99.97 | 99.99 | 99.97 | 99.50 | 90.38 |
| Logistic Regression | 99.92 | 99.98 | 99.87 | 99.40 | 69.23 |
| Naïve Bayes | 93.66 | 96.43 | 87.08 | 42.71 | 90.38 |

TABLE VIII
PERFORMANCE OF MULTINOMIAL MACHINE LEARNING MODELS ON THE NSL-KDD TEST DATASET.

| | | Detection Rate [%] | | | | Runtime [sec] | |
|---|---|---|---|---|---|---|---|
| **Model** | **Accuracy [%]** | **DoS** | **Probe** | **R2L** | **U2R** | **Training** | **Test** |
| Deep Learning H2O | 84.13 | 94.57 | 86.66 | 56.26 | 29.85 | 188.81 | 0.05 |
| Deep Learning 4J | 71.36 | 84.66 | 67.20 | 41.07 | 44.78 | 368.98 | 1.80 |
| LibSVM | 57.69 | 69.38 | 91.86 | 0.07 | 1.49 | 1004.31 | 43.01 |
| Random Forest | 76.92 | 88.35 | 88.76 | 37.75 | 62.69 | 29.02 | 0.50 |
| Logistic Regression | 85.54 | 97.53 | 77.94 | 61.66 | 52.24 | 133.59 | 0.39 |
| Naïve Bayes | 71.11 | 75.23 | 90.91 | 43.95 | 67.16 | 0.44 | 1.25 |



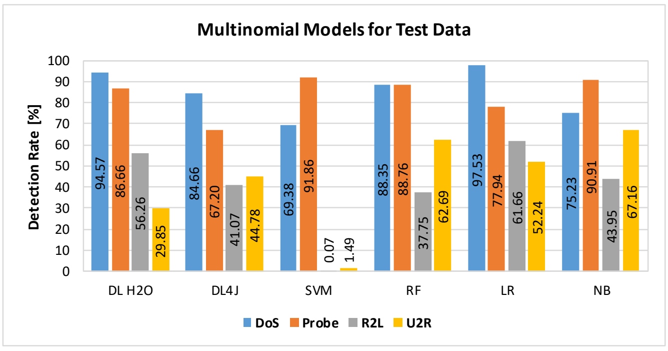


Fig. 5. Detection rates of multinomial models for the: DoS, Probe, R2L and U2R intrusion classes.

Regression and Deep Learning H2O at around 85% accuracy, while the weakest being LibSVM at close to 58%.

Fig. 4 illustrates the results in terms of detection rate, showing that no single model can best and consistently predict the different attack classes. For cross-validation approach, the top model would be Random Forest which achieves over 99% detection rate for DoS, Probe and R2L and 90% for U2R. The Deep Learning models perform well for the first three attack classes but for probe H2O and DL4J only achieve 65% and and 44% respectively.

The results are even more varied for the train plus test approach, where Logistic Regression has highest detection rate for DoS and R2L, SVM for Probe, and Naïve Bayes for U2R class. Moreover, while H2O continues to outperform DL4J for the first three classes, DL4J achieves higher detection rate for U2R. The conclusion that can be drawn is that using one model alone may not be sufficient and ensemble methods that combine several different machine learning algorithms may be more suitable for predicting the intrusion class.

### *C. Comparison with Previous Works*

It is relevant to compare the results of this research study with the previous work by Niyaz et al. [20]. The H20 Deep Learning binomial model achieved 83.87% accuracy on the NSL-KDD test dataset, which is higher than the 78.06% value of the deep learning model based on soft-max regression (SMR) but less than the 88.39% value of the self-taught learning (STL) from [20]. For multinomial intrusion class detection on the test dataset, the SMR and STL deep learning algorithms achieved an accuracy of 75.23% and 79.10% respectively. These values are lower than the 83.87% achieved by the H2O model evaluated in this study. However, the comparison is not very direct as we built 4-class multinomial models only on intrusion data excluding normal traffic, while Niyaz at al. built 5-class multinomial models on the complete dataset including the normal traffic.

## VI. Conclusions

While previous research studies have applied machine learning for network intrusion detection, the capabilities of deep learning for real time detection has not been thoroughly investigated. This paper proposed a cloud based prototype system that integrates two deep learning models: one binomial classification model to identify if there is an intrusion or not, and a second multinomial model to detect the attack class in case of an intrusion.

An evaluation study was carried out using the benchmark NSL-KDD dataset which consists of normal traffic record as well as intrusions grouped into four classes U2R, Probe, R2L and U2R. The evaluation study compared deep learning models built using H2O and DeepLearning4J libraries, with other commonly used machine learning models such as SVM, Random Forest, Naïve Bayes and Logistic Regression. The results showed that the choice of the deep learning library is an important factor to consider for real-time applications, as using default settings H2O outperformed DL4J in terms of accuracy and detection rates, and was faster to train and build the models. In terms of accuracy the H2O deep learning based binomial and multinomial models also outperformed the other machine learning models. The H2O binomial model also provided better detection rates than the other models. However, for multinomial classification no model provided consistent and best performance for all intrusion classes.

Future work directions include investigation of ensemble approaches that combine multiple machine learning methods, in order to improve the detection rates of multinomial classifiers. One limitation of this study is the use of the NSL-KDD dataset, thus other future work directions would be to use datasets with other intrusion types and/or real-time network traffic metrics.

## References

[1] Ericsson, "Internet of Things forecast," Nov. 2016. [Online]. Available: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast

[2] D. Marsh, "Are Ethical Hackers the Best Solution for Combating the Growing World of Cyber-Crime?" Thesis, University Honors College, Middle Tennessee State University, May 2017. [Online]. Available: http://jewlscholar.mtsu.edu/xmlui/handle/mtsu/5256

[3] T. Armerding, "The 17 biggest data breaches of the 21st century," Jan. 2018. [Online]. Available: https://www.csoonline.com/article/2130877/data-breach/the-biggest-data-breaches-of-the-21st-century.html

[4] L. Wang and R. Jones, "Big Data Analytics for Network Intrusion Detection: A Survey," *International Journal of Networks and Communications*, vol. 7, no. 1, pp. 24–31, 2017.

[5] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in Cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, Jan. 2013.

[6] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid Intelligent Intrusion Detection Scheme," in *Soft Computing in Industrial Applications*, ser. Advances in Intelligent and Soft Computing. Springer, Berlin, Heidelberg, 2011, pp. 293–303.

[9] 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 (CISDA)*, Jul. 2009, pp. 1–6.

[7] L. N. Smith, "Best Practices for Applying Deep Learning to Novel Applications," *arXiv:1704.01568 [cs]*, Apr. 2017, arXiv: 1704.01568. [Online]. Available: http://arxiv.org/abs/1704.01568

[8] "H2O.ai." [Online]. Available: https://www.h2o.ai/

[10] B. Cui and S. He, "Anomaly Detection Model Based on Hadoop Platform and Weka Interface," in *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Jul. 2016, pp. 84–89.

[11] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi, "Real-time network anomaly detection system using machine learning," in *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, Mar. 2015, pp. 267–270.

[12] V. Timčenko and S. Gajin, "Ensemble classifiers for supervised anomaly based network intrusion detection," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, Sep. 2017, pp. 13–19.

[13] S. Manohar Naik and N. Geethanjali, "A Multi-Fusion Pattern Matching Algorithm for Signature-Based Network Intrusion Detection System," Aug. 2016. [Online]. Available: http://www.preprints.org/manuscript/201608.0197/v1

[14] P. Jongsuebsuk, N. Wattanapongsakorn, and C. Charnsripinyo, "Real-time intrusion detection with fuzzy genetic algorithm," in *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, May 2013, pp. 1–6.

[15] M. M. Rathore, A. Paul, A. Ahmad, S. Rho, M. Imran, and M. Guizani, "Hadoop Based Real-Time Intrusion Detection for High-Speed Networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6.

[16] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feb. 2017, pp. 313–316.

[17] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, Oct. 2015, pp. 11–20.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[19] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," *International Journal of Security and Its Applications*, vol. 9, no. 5, pp. 205–216, 2015.

[20] Q. Niyaz, W. Sun, A. Javaid, and M. 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)*, ser. BICT'15. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.

[21] R. C. Aygun and A. G. Yavuz, "Network Anomaly Detection with Stochastically Improved Autoencoder Based Models," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, Jun. 2017, pp. 193–198.

[22] C. Shearer, "The CRISP-DM model: the new blueprint for data mining," *Journal of Data Warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[23] "KDD Cup 1999 Data Set." [Online]. Available: http://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data

[24] "The NSL-KDD Data Set." [Online]. Available: https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/

[25] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA dataset for intrusion detection system evaluation," in *Proc. SPIE 6973, Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*, vol. 6973. International Society for Optics and Photonics, Mar. 2008, p. 69730G.

[26] A. Arora, A. Candel, J. Lanford, E. LeDell, and V. Parmar, "Deep Learning with H2O," 2015. [Online]. Available: http://h2o.ai/resources

[27] "Deeplearning4j: Open-source, Distributed Deep Learning for the JVM." [Online]. Available: https://deeplearning4j.org/

[28] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java." [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/