Deep Learning Approach for Intelligent Intrusion Detection System

K. Sushma[1], Meghana Mandala[2], Rohith Gunda[2], Anvesh Jvn[2], Mali Sai Chand Reddy[2]

[1,2]Department of Information Technology

[1,2] CMR Engineering College, Kandlakoya, Medchal, Hyderabad.

ABSTRACT

The Industrial Internet of Things has grown significantly in recent years. While implementing industrial digitalization, automation, and intelligence introduced a slew of cyber risks, the complex and varied industrial Internet of Things environment provided a new attack surface for network attackers. As a result, conventional intrusion detection technology cannot satisfy the network threat discovery requirements in today's Industrial Internet of Things environment.

An intrusion detection system (IDS) is a critical component of network security protection because it enables the system to detect network intrusions efficiently. However, in recent years, as the operating environment and structure of the Industrial Internet of Things have changed, traditional intrusion detection models (such as intrusion detection models based on simple machine learning) have been unable to provide adaptive detection, response, and defence against complex network attacks.

Machine learning nowadays is a developing topic; its applications are wide. We can forecast the future through machine learning and classify the right class. Unsupervised solutions do reduce computational complexities and manual support for labeling data but current unsupervised solutions do not consider spatio-temporal correlations in traffic data. To address this, in the existing basic convolutional autoencoder methods are presented. However, the existing autoencoders have lot of issues, such as Insufficient training data, training the wrong use case, too lossy, imperfect decoding, misunderstanding important variables, better alternatives, algorithms become too specialized, bottleneck layer is too narrow.

So, to overcome these drawbacks, this work presented the deep learning convolutional network-based intrusion detection framework. The simulations will conduct on UNSW-NB15 dataset, which contains attack and normal classes of data. Initially, the dataset preprocessing operation will perform to remove the missing symbols, unknown characters. Then, the deep learning model applied to perform the training of dataset, which also predicts the normal and attack class from test data.

Keywords: Intrusion detection, deep learning, IoT.

1. INTRODUCTION

Information and communications technology (ICT) systems and networks handle various sensitive user data that are prone to various attacks from both internal and external intruders. These attacks can be manual, machine generated, diverse and are gradually advancing in obfuscations resulting in undetected data breaches. For instance, the Yahoo data breach had caused a loss of $350M and Bitcoin breach resulted in a rough estimate of $70M loss. Such cyberattacks are constantly evolving with very sophisticated algorithms with the advancement of hardware, software, and network topologies including the recent developments in the Internet of Things (IoT). Malicious cyber-attacks pose serious security issues that demand the need for a novel, flexible and more reliable intrusion detection system (IDS). An IDS is a proactive intrusion detection tool used to detect and classify intrusions, attacks, or violations of the security policies automatically at network-level and host-level infrastructure in a timely manner. Based on intrusive behaviors, intrusion detection is classified into

network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS). An IDS system which uses network behaviour is called NIDS. The network behaviors are collected using network equipment via mirroring by networking devices, such as switches, routers, and network taps and analyzed in order to identify attacks and possible threats concealed within network traffic. An IDS system which uses system activities in the form of various log files running on the local host computer in order to detect attacks is called HIDS. The log files are collected via local sensors. While NIDS inspects each packet contents in network traffic flows, HIDS relies on the information of log files which includes sensors logs, system logs, software logs, file systems, disk resources, users account information and others of each system.

Many organizations use a hybrid of both NIDS and HIDS. Analysis of network traffic flows is done using misuse detection, anomaly detection and stateful protocol analysis. Misuse detection uses predefined signatures and filters to detect the attacks. It relies on human inputs to constantly update the signature database. This method is accurate in finding the known attacks but is completely ineffective in the case of unknown attacks. Anomaly detection uses heuristic mechanisms to find unknown malicious activities. In most of the scenarios, anomaly detection produces a high false positive rate. To combat this problem, most organizations use the combination of both misuse and anomaly detection in their commercial solution systems. Stateful protocol analysis is most powerful in comparison to the detection methods due to the fact that stateful protocol analysis acts on the network layer, application layer and transport layer. This uses the predefined vendors specification settings to detect the deviations of appropriate protocols and applications.

Though deep learning approaches are being considered more recently to enhance the intelligence of such intrusion detection techniques, there is a lack of study to benchmark such machine learning algorithms with publicly available datasets. The most common issues in the existing solutions based on machine learning models are: firstly, the models produce high false positive rate with wider range of attacks; secondly, the models are not generalizable as existing studies have mainly used only a single dataset to report the performance of the machine learning model; thirdly, the models studied so far have completely unseen today's huge network traffic; and finally the solutions are required to persevere today's rapidly increasing high-speed network size, speed and dynamics. These challenges form the prime motivation for this work with a research focus on evaluating the efficacy of various classical machine learning classifiers and deep neural networks (DNNs) applied to NIDS and HIDS. Overall, this work has made the following contributions to the cyber security domain:

- By combining both NIDS and HIDS collaboratively, an effective deep learning approach is proposed by modelling a deep neural network (DNN) to detect cyberattacks proactively. In this study, the efficacy of various classical machine learning algorithms and DNNs are evaluated on various NIDS and HIDS datasets in identifying whether network traffic behaviour is either normal or abnormal due to an attack that can be classified into corresponding attack categories.
- The advanced text representation methods of natural language processing (NLP) are explored with host-level events, i.e., system calls with the aim to capture the contextual and semantic similarity and to preserve the sequence information of system calls. The comparative performance of these methods is conducted with the NSL-KDD dataset.
- This study uses various benchmark datasets to conduct a comparative experimentation. This is mainly due to the reason that each data set suffers from various issues such as data corruptions, traffic variety, inconsistencies, out of date and contemporary attacks.

2. LITERATURE SURVEY

A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection.

AUTHORS: Mishra .P, Varadharajan. V, Tupakula. U and Pilli E. S

The increasing rate of intrusions in the network and host machines have badly affected the security and privacy of users. Researchers have extensively worked on various solutions to detect intrusions. The security aspects of intrusion detection using machine learning approach have been considered in our paper. We have described various types of attacks in the network and host systems with the brief description of their attack features. The analysis performed, reveals that if a technique is performing well for detecting an attack, it may not perform same for detecting other attacks. Hence, the relevance of a technique for specific attacks has been presented by classifying various machine learning techniques for each type of attack. The critical performance analysis of various machine learning algorithms has been done in an evolutionary way. The comparison has been carried out with single classifier approaches and multiple classifier approaches. The influence of a classifier with other classifier is not only analyzed but also the influence of a feature subset with the classifier is analyzed. We have shown that even if an optimal feature set is sufficient for analyzing the behavior of an attack, it is not good for analyzing the behavior of other attacks.

Use of Data Visualization for Zero-Day Malware Detection.

AUTHORS: Venkatraman. S, Alazab. M

This paper proposed a new hybrid method of feature based and data-based visualization of similarity mining to identify and classify malware accurately. Our visualization technique is effectively used to compare malware samples for better communication of their behaviour patterns and faster detection and classification of new malware (zero-day malware). We calculated the similarities between the malware variants using eight different distance measures to generate similarity matrices and to identify the malware family by adopting visualization of the distance scores. The experimental study of our proposed method involved large datasets of about 75,000 samples with more than two-thirds consisting of malware samples and benign samples forming the rest. By performing similarity mining of the innumerable obfuscations of extended x86 IA-32 (opcodes) found in these malware samples, we were successfully able to detect and classify unknown malware that had escaped from traditional detection methods. Te proposed method is efficient and accurate in identifying malware visually due to three main properties observed through our experimental results: (1) Malware opcodes exhibit significant dissimilarity of behaviour patterns as compared to the benign opcodes and hence result in very high true positives (2) For malware programs belonging to the same family, the uniqueness and closeness in similarity can be visually deciphered through the colour-coded distance measures of the similarity matrix and each malware family exhibits a unique visual pattern of the similarity matrix.

Machine Learning Based Botnet Identification Traffic.

AUTHORS: Ahmad Azab, Mamoun Alazab and Mahdi Aiash

Botnet is currently the main threat in the internet against individuals and organizations, causing large losses for both parties. Botnet C&C channel traffic identification is a vital task to treat the infected devices, take C&C server down and track down cybercriminals. Researchers proposed approaches as DPI, DNS request behavior, temporal, correlation and machine learning to detect the C&C channel traffic. The approaches addressed the solutions in terms of online classification capability by monitoring a few packets in a flow, supporting various transport and application protocols as TCP,

UDP, HTTP and IRC, avoiding accessing packet's content, relying on a single phase's traffic for the detection process by monitoring only C&C channel traffic, monitoring a single device's network traffic and detecting untrained versions for the targeted application by analyzing and building the classifier on a different single version. None of the approaches fulfilled these characteristics in a single solution. The novel methodology, CONIFA, was used to overcome this gap, by fulfilling all the aforementioned characteristics in a single solution. CONIFA mainly relies on the machine learning approach and aims in filling its gap by the fact that the machine learning approach performance does degrade if the untrained versions have statistical values that are dissimilar from the one used by the built classifier. CONIFA deploys cost sensitive algorithms and different feature combinations concepts. These two concepts are used to maximize the detection of the trained version, thus increasing the likelihood in detecting the untrained version.

Disclosure of Cyber Security Vulnerabilities: Time Series Modelling

AUTHORS: Tang.M,Alazab.M, Luo.Y and Donlon.M

Computer systems are vulnerable to cyber-attack from both inside and outside of the system network. During the process of producing software products and website design, vendors and developers unintentionally create vulnerabilities that can be exploited later by cyber criminals. The continued growth of the Internet has resulted in the increasing sophistication of tools and methods for identification and prevention of new vulnerabilities during the development lifecycle of the emerging inter-connected systems, such as networked and intelligent cars and industrial control systems. In this paper, we seek to minimise the potential harm caused by the exploitation of disclosed vulnerabilities. We particularly tackled a challenging real-world cyber security problem in the domain of discovering and modelling vulnerability disclosure trend. Weproposed a novel and rigorous statistical framework towards deepening our understanding about the disclosure dynamics. By leveraging the proposed framework, we formally discovered the existence of volatility clustering (ARCH effect) in our case study on long-term NVD data. We also thoroughly studied the persistence of ARCH effect by utilising different GARCH models.

Machine Learning and Deep Learning Methods for Cybersecurity

AUTHORS: Yang Xi, Lingshuang Kong and Yuling Chen

This paper presents a literature review of ML and DL methods for network security. The paper, which has mostly focused on the last three years, introduces the latest applications of ML and DL in the field of intrusion detection. Unfortunately, the most effective method of intrusion detection has not yet been established. Each approach to implementing an intrusion detection system has its own advantages and disadvantages, a point apparent from the discussion of comparisons among the various methods. Thus, it is difficult to choose a particular method to implement an intrusion detection system over the others. Datasets for network intrusion detection are very important for training and testing systems. The ML and DL methods do not work without representative data, and obtaining such a dataset is difficult and time-consuming.

Host based intrusion detection system using frequency analysis of n-gram terms.

AUTHORS: Subba. B, Biswas.S & Karmakar. S

In this paper, we proposed a HIDS framework for detecting intrusive system processes based on frequency analysis of n-gram terms in the system call trace files. The proposed framework initially transforms the system call traces to a ngram vector representation model. It then uses a dimensionality reduction process to reduce the size of the input feature vectors by selecting only those n-gram terms,

whose frequencies are greater than the predefined threshold value. Finally, the dimensionality reduced vectors are processed by different classifiers (Naive Bayes, SVM, MLP and C4.5 Decision Tree) to determine whether the corresponding system call traces are normal and intrusive. Experimental results on the benchmark ADFA-LD dataset show that the proposed framework accurately identifies the normal and intrusive system processes, while at the same time minimizes the overall computational overhead of the HIDS. For our future work, we aim to improve and fine tune various parameters of the proposed framework to further enhance its performance.

Random forests-based network intrusion detection systems

AUTHORS: Zhang, Jiong, Mohammad Zulkernine, and Anwar Haque

In this paper, we outline three data-mining-based frameworks for network intrusion detection. We apply the random forests algorithm in misuse, anomaly, and hybrid detection. To address the problems of rule-based systems, we employ the random forests algorithm to build patterns of intrusions. By learning over training data, the random forests algorithm can build the patterns automatically instead of coding rules manually. The proposed approaches are implemented in Java program using the WEKA environment [5] and the Fortran 77 program [2]. We evaluate the implementations over different datasets obtained from the KDD'99 datasets, and the experimental results show that the performances of our approaches are better than the best KDD'99 results. In our misuse detection framework, patterns of intrusions are built in the offline phase, and the system can automatically detect intrusions in real time using the built patterns. To improve the accuracy of the system, we use the feature selection algorithm and optimize the parameters of the random forests algorithm.

AdaBoost-Based Algorithm for Network Intrusion Detection

AUTHORS: Hu, Weiming, Wei Hu, and Steve Maybank.

We have proposed an AdaBoost-based algorithm for intrusion detection. In the algorithm, decision stumps are used as weak classifiers. The decision rules are provided for both categorical and continuous features. The relations between categorical and continuous features are handled naturally, without any forced conversions between these two types of features. A simple overfitting handling is used to improve the learning results. In the specific case of network intrusion detection, we use adaptable initial weights to make the tradeoff between the detection and false-alarm rates.

A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks

AUTHORS: Yin. C, Zhu. Y, Fei. J & He.X

The RNN-IDS model not only has a strong modelling ability for intrusion detection, but also has high accuracy in both binary and multiclass classification. Compared with traditional classification methods, such as J48, naive bayesian, and random forest, the performance obtains a higher accuracy rate and detection rate with a low false positive rate, especially under the task of multiclass classification on the NSL-KDD dataset. The model can effectively improve both the accuracy of intrusion detection and the ability to recognize the intrusion type. Of course, in the future research, we will still pay attention to reduce the training time using GPU acceleration, avoid exploding and vanishing gradients, and study the classification performance of LSTM, Bidirectional RNNs algorithm in the field of intrusion detection.

Enhanced Network Anomaly Detection Based on Deep Neural Networks

AUTHORS: Naseer. S, Saleem. Y, Khalid. S, Bashir. M. K, Han. J, Iqbal. M. M & Han. K

In this paper, Intrusion Detection models were proposed, implemented and trained using different deep neural network architectures including Convolutional Neural Networks, Autoencoders, and Recurrent Neural Networks. These deep models were trained on NSLKDD training dataset and evaluated on both test datasets provided by NSLKDD namely NSLKDDTest+ and NSLKDDTest21. For training and evaluation of deep models, a GPU powered test-bed using keras with theano backend was employed. To make model comparisons more credible, we implemented conventional ML IDS models with different well-known classification techniques including Extreme Learning Machine, k-NN, Decision-Tree, RandomForest, Support Vector Machine, Naive-Bays, and QDA. Both DNN and conventional ML models were evaluated using wellknown classification metrics including RoC Curve, Area under RoC, Precision-Recall Curve, mean average precision and accuracy of classification. Both DCNN and LSTM models showed exceptional performance with 85% and 89% Accuracy on test dataset which demonstrates the fact that Deep learningis not only viable but rather promising technology for information security applications like other application domains. Our future research will be directed towards investigating deep learning as feature extraction tool to learn efficient data representations for anomaly detection problem.

3. PROPOSED SYSTEM

Machine learning techniques are being widely used to develop an intrusion detection system (IDS) for detecting and classifying cyberattacks at the network-level and the host-level in a timely and automatic manner. However, many challenges arise since malicious attacks are continually changing and are occurring in very large volumes requiring a scalable solution. There are different Intrusion datasets available publicly for further research by cyber security community. However, no existing study has shown the detailed analysis of the performance of various machine learning algorithms on various publicly available datasets. Due to the dynamic nature of Intrusion with continuously changing attacking methods, the Intrusion datasets available publicly are to be updated systematically and benchmarked. This work evaluates the performance of various classical algorithms such as SVM, Random Forest and Deep Neural Network (DNN) etc to detect attacks on network using KDD, NSL datasets. The existing classical algorithms (SVM, Random Forest) are unable to predict dynamic (if attacker introduce new attacks with changes in attack parameter) attacks and needs to be trained in advance.

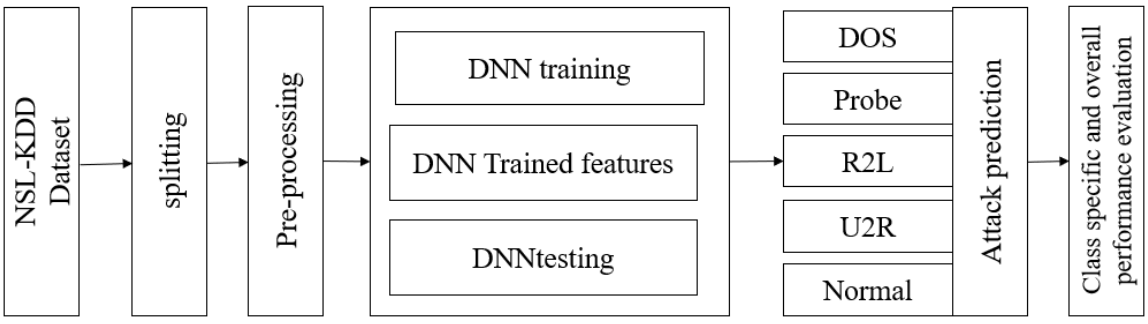


Fig. 1: Proposed IDS methodology.

Figure 1 shows the block diagram of proposed method. Initially, NSL-KDD dataset is split into 80% for training and 20% for testing. Then, dataset preprocessing operation is performed to normalize the entire dataset. Further, DNN classifier is used for prediction of attacks from test sample. The performance evaluation is carried out to show supremacy of the proposed method. The DNN is a famous algorithm which has high predicting ratio in all fields such as image processing, data classification etc. Therefore, DNN model is capable of detecting such attacks and to overcome from

these attacking problems with dynamic attack signatures. The proposed DNN model contains multiple number of layers. The DNN algorithm keeps filtering training algorithm with hidden layer to form most accurate model to predict testing class. The common classes are Normal, Remote to user (R2L), Denial-of-Service (DOS), User to Root (U2R), Probe but in dataset we have other names, but all those names come under these classes.

3.1 Dataset

NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD'99 data set. Although, this new version of the KDD data set still suffers from some of the problems discussed by McHugh and may not be a perfect representative of existing real networks, because of the lack of public data sets for network based IDSs, we believe it still can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods.

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

Data files

- KDDTrain+.ARFF: The full NSL-KDD train set with binary labels in ARFF format.
- KDDTrain+.TXT: The full NSL-KDD train set includes attack-type labels and difficulty level in CSV format.
- KDDTrain+_20Percent.ARFF: A 20% subset of the KDDTrain+.arff file
- KDDTrain+_20Percent.TXT: A 20% subset of the KDDTrain+.txt file
- KDDTest+.ARFF: The full NSL-KDD test set with binary labels in ARFF format
- KDDTest+.TXT: The full NSL-KDD test set includes attack-type labels and difficulty level in CSV format.
- KDDTest-21.ARFF: A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21
- KDDTest-21.TXT: A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21.

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

- It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
- There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
- The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.

- The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

3.2 Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data preprocessing tasks.

Need of Data Preprocessing: A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

3.3 Splitting the Dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

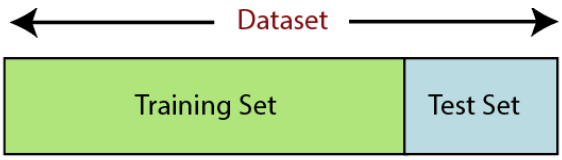


Fig. 2: Splitting the dataset.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

3.4 DNN

A typical architecture of DNN model for intrusion recognition is shown in Figure 4.5. DNNs are generally composed of three parts. Dense layer for feature extraction. The convergence layer, also known as the pooling layer, is mainly used for feature selection. The number of parameters is reduced by reducing the number of features. The full connection layer carries out the summary and output of the characteristics. A dense layer is consisting of a dense process and a nonlinear activation function ReLU.

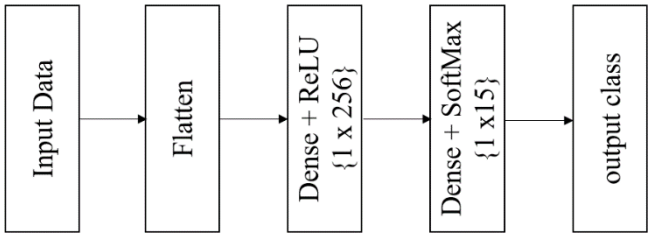


Fig. 3: Proposed DNN model.

The leftmost data is the input layer, which the computer understands as the input of several matrices. Next is the dense layer, the activation function of which uses ReLU. The pooling layer has no activation function. The combination of dense and pooling layers can be constructed many times. The combination of dense layer and dense layer or dense layer and pool layer can be very flexibly, which is not limited when constructing the model. But the most common DNN is a combination of several dense layers and pooling layers. Finally, there is a full connection layer, which acts as a classifier and maps the learned feature representation to the sample label space.

DNN mainly solves the following two problems.

1) Problem of too many parameters: It is assumed that the size of the input test data is $50 * 50 * 3$. If placed in a fully connected feedforward network, there are 7500 mutually independent links to the hidden layer. And each link also corresponds to its unique weight parameter. With the increase of the number of layers, the size of the parameters also increases significantly. On the one hand, it will easily lead to the occurrence of over-fitting phenomenon. On the other hand, the neural network is too complex, which will seriously affect the training efficiency. In DNNs, the parameter sharing mechanism makes the same parameters used in multiple functions of a model, and each element of the denseal kernel will act on a specific position of each local input. The neural network only needs to learn a set of parameters and does not need to optimize learning for each parameter of each position.

2) Data stability: Data stability is the local invariant feature, which means that the natural data will not be affected by the scaling, translation, and rotation of the data size. Because in deep learning, data enhancement is generally needed to improve performance, and fully connected feedforward neural is difficult to ensure the local invariance of the data. This problem can be solved by dense operation in DNN.

ReLU layer: Networks those utilizes the rectifier operation for the hidden layers are cited as ReLU. This ReLU function $\mathcal{G}(\cdot)$ is a simple computation that returns the value given as input directly if the

value of input is greater than zero else returns zero. This can be represented as mathematically using the function $max(\cdot)$ over the set of 0 and the input x as follows:

$$\mathcal{G}(x) = \max\{0, x\}$$

SoftMax classifier: Generally, SoftMax function is added at the end of the output since it is the place where the nodes are meet finally and thus, they can be classified as shown in Figure 4.5. Here, X is the input of all the models and the layers between X and Y are the hidden layers and the data is passed from X to all the layers and Received by Y. Suppose, we have 10 classes, and we predict for which class the given input belongs to. So, for this what we do is allot each class with a particular predicted output. Which means that we have 10 outputs corresponding to 10 different class and predict the class by the highest probability it has.

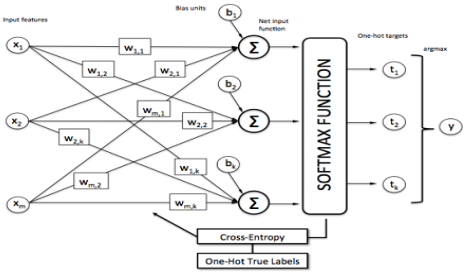


Fig. 4: Intrusion class prediction using SoftMax classifier.

In Figure 4, and we must predict what is the object that is present in the test data. In the normal case, we predict whether the Intrusion is A. But in this case, we must predict what is the object that is present in the test data. This is the place where SoftMax comes in handy. As the model is already trained on some data. So, as soon as the test data is given, the model processes the test data, send it to the hidden layers and then finally send to SoftMax for classifying the test data. The SoftMax uses a One-Hot encoding Technique to calculate the cross-entropy loss and get the max. One-Hot Encoding is the technique that is used to categorize the data. In the previous example, if SoftMax predicts that the object is class A then the One-Hot Encoding for:

Class A will be [1 0 0]

Class B will be [0 1 0]

Class C will be [0 0 1]

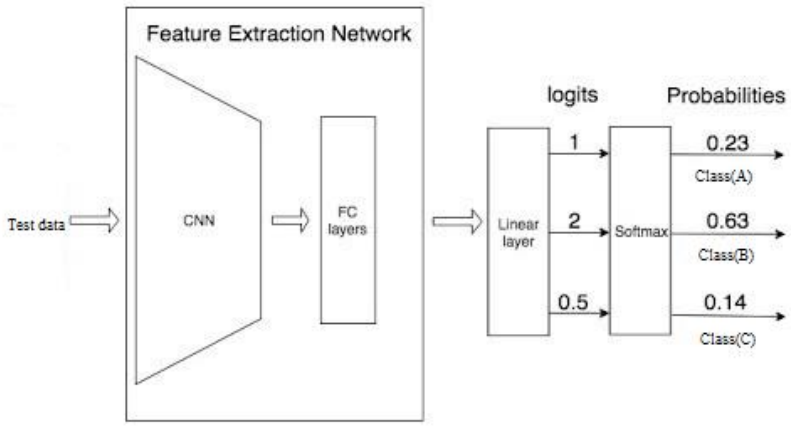


Fig. 5: Example of SoftMax classifier.

From the Fig. 6, we see that the predictions are occurred. But generally, we don't know the predictions. But the machine must choose the correct predicted object. So, for machine to identify an object correctly, it uses a function called cross-entropy function. So, we choose more similar value by using the below cross-entropy formula.

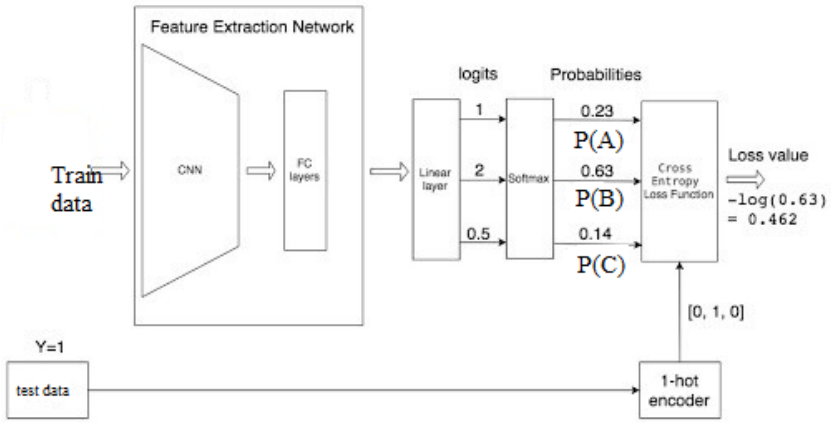


Fig. 6: Example of SoftMax classifier with test data.

In the above example we see that 0.462 is the loss of the function for class specific classifier. In the same way, we find loss for remaining classifiers. The lowest the loss function, the better the prediction is. The mathematical representation for loss function can be represented as: -

$$LOSS = np.sum(-Y * np.log(Y\_pred))$$

4. RESULT AND DISCUSSION

This project evaluates the performance of various classical algorithms such as SVM, and Random Forest to detect attacks on network using IDS datasets such as KDD, NSL but these classical algorithms unable to predict dynamic (if attacker introduce new attacks with changes in attack parameter) attacks and need to be trained in advance to detect such attacks. To overcome this problem, this work evaluates performance of customized neural network (CNN) model with dynamic attack signatures. The obtained detection accuracy of CNN shown to be better as compared to all classical algorithms.

Here to implement this concept, KDD and NSL dataset combination is used with SVM, Random Forest and CNN model. CNN models keep filtering training algorithms with hidden layers to form the most accurate model to predict testing class. It is a famous model which has a high predicting ratio in all fields such as image processing, data classification etc.

Below are the column names of dataset.

duration,protocol_type,service,flag,src_bytes,dst_bytes,land,wrong_fragment,urgent,hot,num_failed_logins,logged_in,num_compromised,root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,is_host_login,is_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate,label

In the above dataset columns label is the name of attacks, all above comma separated names in bold format are the names of request signature.

Below are the values of the above dataset columns.

0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1,1,0,0,0.05,0.06,0,255,9,0.04,0.05,0,0,1,1,0,0,Neptune

The above two records are the signature values and the last value contains class label such as normal request signature or attack signature. In the second record 'Neptune' is the name of attack. Similarly in dataset you can find nearly 30 different names of attacks.

In the above dataset records we can see some values are in string format such as tcp, ftp_data and these values are not important for prediction and these values will be remove out by applying PREPROCESSING Concept. All attack names will not be identified by algorithm if it's given in string format, so we need to assign numeric value for each attack. All this will be done in PREPROCESS steps and then a new file will be generated called 'clean.txt' which will be used to generate training model.

In below line i am assigning numeric id to each attack

"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"satan":7,"smurf":8,"pod":9,"back":10,"guess_passwd":11,"ftp_write":12,"multihop":13,"rootkit":14,"buffer_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptunnel":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":35,"xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39

In the above lines we can see normal is having id 0 and Neptune 1 and goes on for all attacks.

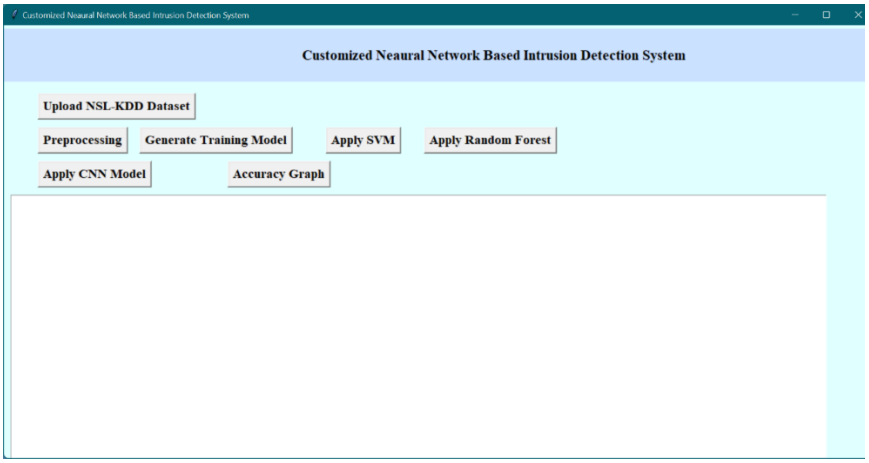
In the paper author describe about Normal, R2L, DOS, U2R, DOS, Probe but in dataset we have other names but all those names comes under 5 categories such as Normal, R2L, DOS, U2R, DOS, Probe.

Attack category	Attack name
Denial of service (DoS)	**Apache2**, Smurf, Neptune, Back, Teardrop, Pod, Land, **Mailbomb**, **Processtable**, **UDPstorm**
Remote to local (R2L)	WarezClient, Guess_Password, WarezMaster, Imap, Ftp_Write, **Named**, MultiHop, Phf, Spy, **Sendmail**, **SnmpGetAttack**, **SnmpGuess**, **Worm**, **Xsnoop**, **Xlock**
User to root (U2R)	Buffer_Overflow, **Httptuneel**, Rootkit, LoadModule, Perl, **Xterm**, **Ps**, **SQLattack**
Probe	Satan, **Saint**, Ipsweep, Portsweep, Nmap, **Mscan**

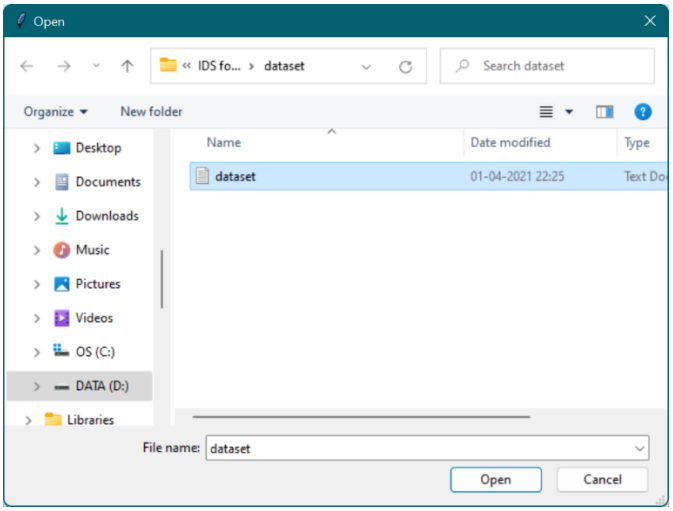
List of attacks presented in NSL-KDD dataset

From the above screen shots we can understand that Neptune attack belongs to DOS category. Similarly other attacks belong to different categories.

UI OUTPUT

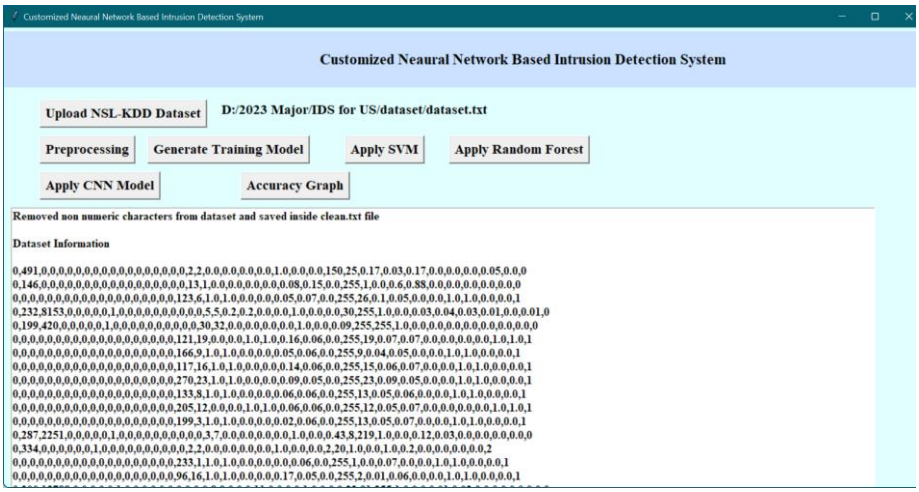


In above screen click on 'Upload NSL-KDD Dataset' to upload dataset.

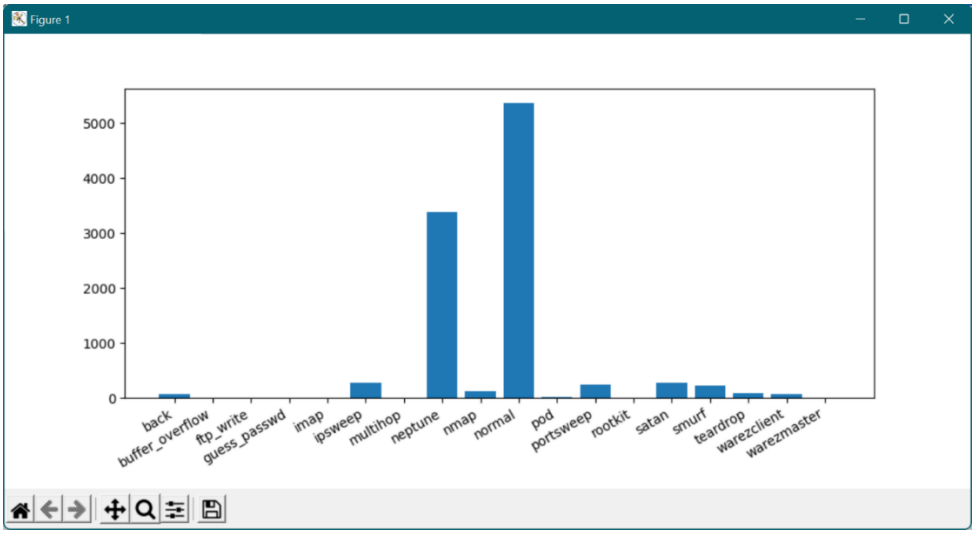


After uploading dataset will get below screen.

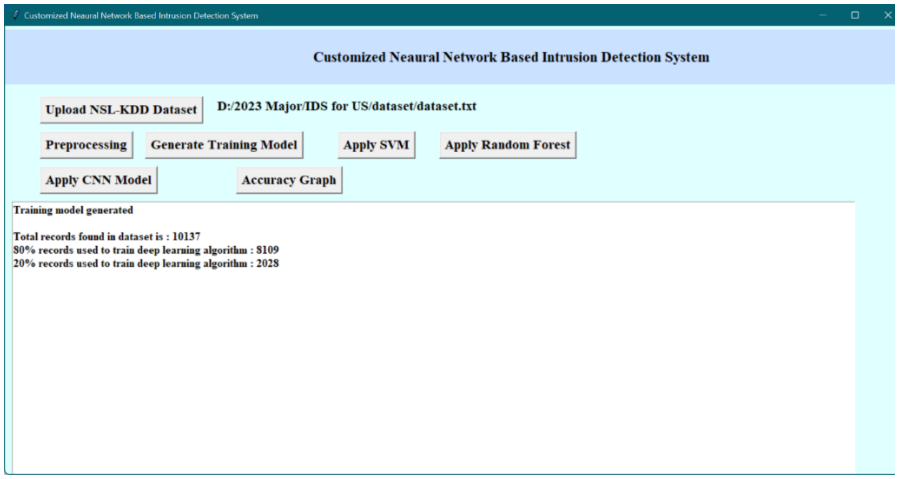
Now click on 'Preprocessing' to assign numeric values to each attack names as algorithms will not understand string names.



In above screen we can see we assign numeric id to each attack and will get below graph which display number of different attacks.

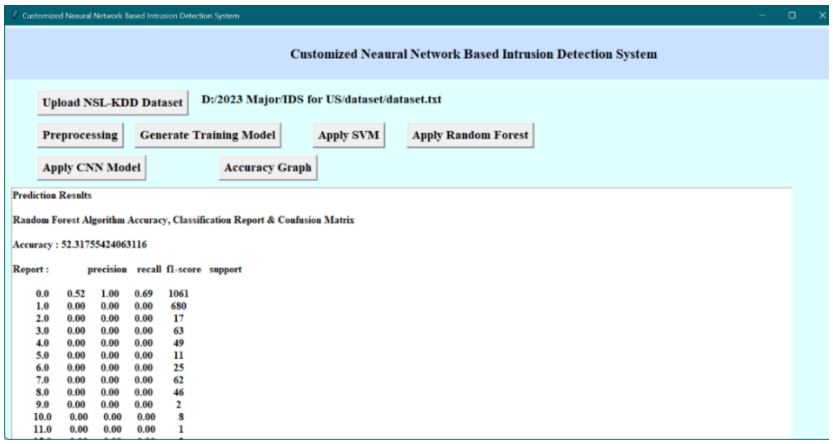


In above graph x-axis represents attack name found in dataset and y-axis represents count of that attack type. Now click on 'Generate Training Model' to split dataset into train and test part where application used 80% dataset to train algorithms and 20% to test algorithms prediction accuracy.

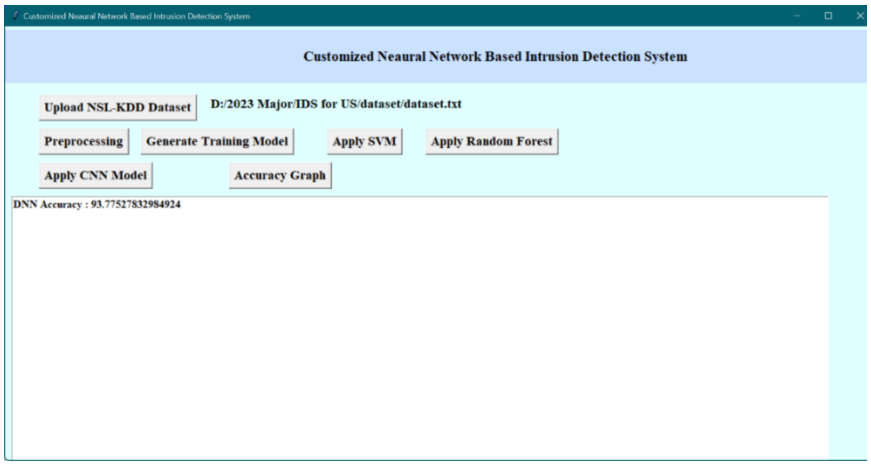


In above screen we can see dataset contains total 10137 records and application using 8109 records for training and 2028 records for testing algorithm prediction accuracy. Now train and test data is ready and click on 'Apply SVM' to get its prediction accuracy.

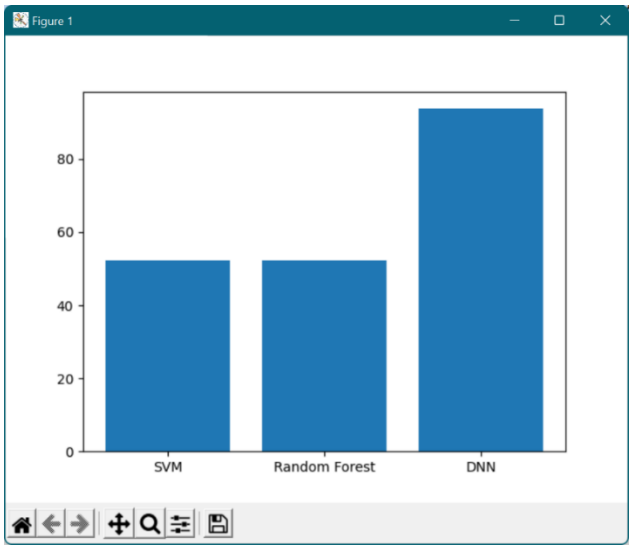
In above screen with SVM we got 52.26% accuracy and scroll down above screen text area to get confusion matrix. Now click on 'Apply Random Forest' to get its accuracy.



In above screen we can see random forest got 52.31% accuracy. Now apply proposed CNN Model.



In above screen we can see DNN accuracy is 93.77% which is better than other two algorithms. Now click on 'Accuracy Graph' to get below graph



In above graph x-axis represents algorithm name and y-axis represents accuracy and DNN is the proposed technique which got high accuracy compared to traditional algorithms such SVM and random forest.

5. CONCLUSION AND FUTURE SCOPE

This project proposed a hybrid intrusion detection system using a highly scalable framework on commodity hardware server which has the capability to analyze the network and host-level activities. The framework employed distributed deep learning model with DNNs for handling and analyzing very large-scale data in real time. The DNN model was chosen by comprehensively evaluating their performance in comparison to classical machine learning classifiers on various benchmark IDS datasets. In addition, we collected host-based and network-based features in real-time and employed the proposed DNN model for detecting attacks and intrusions. In all the cases, we observed that DNNs exceeded in performance when compared to the classical machine learning classifiers. Our proposed architecture is able to perform better than previously implemented classical machine learning classifiers in both HIDS and NIDS. To the best of our knowledge this is the only framework which has the capability to collect network-level and host-level activities in a distributed manner using DNNs to detect attacks more accurately. The performance of the proposed framework can be further enhanced by adding a module for monitoring the DNS and BGP events in the networks. The execution time of the proposed system can be enhanced by adding more nodes to the existing cluster.

In addition, the proposed system does not give detailed information on the structure and characteristics of the malware. Overall, the performance can be further improved by training complex DNNs architectures on advanced hardware through distributed approach. Due to extensive computational cost associated with complex DNNs architectures, they were not trained in this research using the benchmark IDS datasets. This will be an important task in an adversarial environment and is considered as one of the significant directions for future work.

REFERENCES

[1] Mukherjee, B., Heberlein, L. T., & Levitt, K. N. (1994). Network intrusion detection. IEEE network, 8(3), 26-41.

[2] Larson, D. (2016). Distributed denial of service attacks-holding back the flood. Network Security, 2016(3), 5-7.

[3] Staudemeyer, R. C. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal, 56(1), 136-154.
[4] Venkatraman, S., Alazab, M. "Use of Data Visualisation for Zero-Day Malware Detection," Security and Communication Networks, vol. 2018, Article ID 1728303, 13 pages, 2018. https://doi.org/10.1155/2018/1728303
[5] Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials.
[6] Azab, A., Alazab, M. & Aiash, M. (2016) "Machine Learning Based Botnet Identification Traffic" The 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (Trustcom 2016), Tianjin, China, 23-26 August, pp. 1788-1794.
[7] Vinayakumar R. (2019, January 19). vinayakumarr/Intrusion-detection v1 (Version v1). Zenodo. http://doi.org/10.5281/zenodo.2544036
[8] Tang, M., Alazab, M., Luo, Y., Donlon, M. (2018) Disclosure of cyber security vulnerabilities: time series modelling, International Journal of Electronic Security and Digital Forensics. Vol. 10, No.3, pp 255 - 275.
[9] V. Paxson. Bro: A system for detecting network intruders in realtime. Computer networks, vol. 31, no. 23, pp. 24352463, 1999. DOI http://dx.doi. org/10.1016/S1389-1286(99)00112-7
[10] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.