Intrusion Detection On CICIDS 2017

Nidhi Sakhala
Purdue University
West Lafayette
nsakhala@purdue.edu

Pooja Patil
Purdue University
West Lafayette
patil42@purdue.edu

ABSTRACT

At a time when most of the information is stored and transferred electronically, it is important to keep the information safe and prevent any attacks that could potentially be disastrous. Thus, there is a pressing need to have an intrusion detection system which identifies the attacks on a system and alerts the user. For doing this, an Intrusion detection system must be trained properly to differentiate attacks from genuine data. In this paper, we aim to use Random Forest Algorithm and Recurrent Neural Network to train our Intrusion Detection Systems to identify 9 types of attacks mentioned in CICIDS 2017 dataset.

KEYWORDS

Deep learning, Machine learning, Intrusion detection, Neural Networks

1 INTRODUCTION

With the growth of computer networking, electronic commerce, and web services, information security has become an issue of serious global concern. This is due to complexity, accessibility and openness of the Internet. As more and more computers get connected to internet, malicious usage, attacks and sabotage are increasing day by day. Attackers use different techniques like password cracking, eavesdropping, and sniffing network traffic to expose user's sensitive information. Although a wide range of security technologies such as information encryption, access control, and firewalls are used to protect network-based systems, no network can be 100% secured, as there are still many undetected intrusions. The most popular way to detect intrusions is by examining the network traffic and distinguishing anomalous traffic from normal traffic. Intrusion detection system can be compared with a burglar alarm. For example, a car is protected by a lock system. However, if a thief breaks the lock system, it is the burglar alarm that detects that the car lock has been broken and alerts the user.

Therefore, an Intrusion detection system (IDS) is a security system that monitors computer systems and network traffic and analyzes that traffic for possible hostile attacks originating from outside the organization, for system misuse, or attacks originating from inside the organization [1].

There are two types of intrusion detection systems: Network based, and Host based. Network based intrusion detection system are deployed at points within the network so that it can monitor the incoming and outgoing traffic to detect intrusions. Host based systems run on computers or devices in the network, which get access to the internet, and on enterprise internet network. There are two major methodologies of Intrusion detection: misuse detection and anomaly detection. Misuse is recognized as a pattern of attack. Misuse detection detects intrusions based on the patterns of known intrusions. Such system monitors the network packets and compares them against the database of known attacks. Anomalies are deviations from normal usage behavior. Anomaly detection detects intrusions based on the observed abnormal activity. However, both these techniques have limitations; anomaly detections tend to raise many false alarms. Unusual but normal activity can be considered as an intrusion by anomaly detection. Detection of anomaly patterns is computationally expensive because of the overhead of maintaining and updating several system profiles because behavior patterns and system usage vary from system to system and from user to user. While misuse detection can be very efficient in detecting the attacks, whose patterns are known to the system, but it is not possible to discover all the possible attacks that could occur [2]. It usually fails in detecting zero- day attacks.

Until now a lot of soft computing approaches have been implemented on intrusion detection. These approaches can be classified as Statistics-based, Pattern-based, Rule-based, State-based and Heuristic-based [3]. Statistics-based approaches focus on predefined mean and standard deviation, and probabilities to identify intrusions. Pattern-based detections focus on detecting known attacks through string matching. State-based methods use finite state machine derived from network behaviors to identify attacks. Heuristic-based approach is inspired by artificial intelligence.

In this paper, we are going to implement Random forest and Recurrent Neural network on the dataset retrieved from Canadian Institute of Cyber Security. This data was captured in July 2017 over a period of 5 days.

The remainder of this paper is organized as follows. Section 2 describes the previous deep learning approaches implemented by the researchers for intrusion detection. Section 3 describes the methodology and gives a brief description of the dataset.

2 LITERATURE REVIEW

Intrusion detection is fundamentally a problem of classification. An IDS must classify a particular network packet as normal or abnormal. Hence most of the previous research concentrates on machine learning techniques used to detect intrusions. [Zhang et al 2008] have implemented random forest for anomaly, misuse and hybrid detection. In misuse detection the authors have used random forest for finding the patterns of the attack and later intrusions are detected by matching the network activity against the patterns. In anomaly detection, random forest is used as outlier detection to discover intrusions. The authors have evaluated their results over KDD'99 dataset. The results of this paper show that hybrid system have better detection rate than anomaly detection and misuse detection. Also, anomaly approach detects some intrusions that are missed by misuse approach.

Many data mining approaches have been proposed for intrusion detection [4;5;6]. [Sangkatsanee et al 2011] implemented different machine learning techniques for intrusion detection. Techniques like Decision Tree, Ripper Rule, Back-Propagation Neural Network, Radial Basis Function Neural Network, Bayesian Network, and Naıve Bayesian model. The experimental results of this paper show that decision trees provide higher detection rates as compared to other algorithms and hence the authors developed a new real time IDS based on decision trees. This new real time IDS succeeded to efficiently detect intrusions with improved speed and lesser CPU consumption and memory. Similarly [(7) Kwon et al 2017] provides a survey on anomaly intrusion detection using Restricted Boltzman, Deep Belief Network, Deep Neural Network, Auto Encoder, Recurrent Neural Network and LSTM. All the above techniques were evaluated on two datasets KDDCup 1999 and NSL-KDD. Authors also test the effectiveness of deep learning techniques using a Fully Connected Network model. The results suggested improved accuracy with FCN model.

The rise in Neural Networks research and the emergence of Deep Learning in the past decade has opened a new workbench. Various Deep Learning algorithms are being developed and used for multiple applications like textual data, image processing and other time-series data. Some researches applied these techniques to network data as well. [8] has implemented intrusion detection system using multi layer feed forward neural network and SVM. These systems were tested on the DARPA dataset. Both of the system resulted into high accuracy. However SVM is only able to perform binary classification whereas intrusion detection requires multiple class identifications. Neural networks works well in classifying different types of attacks. For this reason we are implementing neural networks. Another implementation of neural network for IDS is given by [13]. In this paper the researchers have implemented RNN for IDS and studied the performance of the model in multiclass and binary classification. Researchers also studied the impact of different number of neurons on the performance of the model. The results show high accuracy as compared to traditional machine learning algorithms. Similarly [9] has implemented reduced size RNN where input features are categorized into basic features, content features, time-based traffic features, and host-based traffic features and the output is the type of attack. This model is evaluated on KDD dataset. The results of this implementation show that the proposed model has improved classification rate specifically for r2L attacks. The model also had better detection rate and cost per example rate compared to machine learning techniques like clustering, KNN, Jordan ANN, SVM etc. Also due to reduced size, training speed and convergence was improved.

While reading about intrusion detection, we observe that most of the research has been done on KDD datasets of 1998 and 1999. As can be seen, this is a very old dataset. Papers published as recently as 2017[13], are still using this dataset, which is 15+ years old. In these years, there has been a major change in the attacks faced by the community. Continuing research on the same old dataset will definitely have shortcomings. This, and many other issues with the dataset have been observed and discussed in the literature [14,15,16]. These papers talk about problems like how the simulated dataset is insufficient to model real-world scenarios as simulations are done in a controlled environment. They also discuss that the classifiers have low accuracy for U2R and R2L attacks because of low data availability. In [15], Shabhnani et al. implemented various classifiers including a multilayer perceptron neural network, an incremental radial basis function neural network, a Gaussian classifier, a K-means classifier, a nearest neighbor classifier, a C4.5 decision tree, a Fuzzy ARTMAP classifier, a Leader cluster, and a Hyper sphere algorithm. Each classifier was trained using the KDD training dataset and tested using the KDD testing dataset. None of the classifiers were able to detect more than 25% of the attack records with acceptable false alarm rates. In [16], the author mentions how the taxonomy of the attack is biased towards the attacker's point of view and may not be correct for evaluation by the victim's point of view. He also says how there is a lack of "unit of analysis" for this dataset.

Due to these problems, efforts have been made to generate better datasets. In [17], Shiravi et al. describe few metrics to generate a benchmark dataset and provide guidelines that the research community should follow to generate more data. They compare earlier datasets - CAIDA, Internet Traffic Archive, LBNL, DARPA-KDD and DEFCON and show how these datasets lack in the given metrics. These authors generate a dataset which follows the metrics and is used widely in other literature.

Continuing these efforts, Iman et al. generated an even extensive database, which addresses the guidelines mentioned in [17] and additionally mentions 11 rules that a benchmark dataset should follow[12]. They generate a dataset following these rules and the dataset contains many more attacks than have been covered by any of the previous public datasets. Hence we believe that using this dataset for intrusion detection will be more appropriate than the earlier ones.

3 METHODOLOGY

In this section, we explain the algorithms that we aim to use for classification of attacks. We aim to identify if the new data is benign data or an attack on the system. The following sections, will explain the dataset, how we will analyze it and how we will evaluate our results. The research question we aim to solve is - 'Do RNNs and Random Forests help to effectively identify the most common network attacks faced in the domain of cyber security?'

3.1 Dataset

The CICIDS2017 Dataset [10] was obtained from the Canadian Institute for Cybersecurity. We have used this dataset as it provides data in real-world packet format (pcap) as well as CSV files. The CSV files include upto 80 features and contains a wide sample of benign data and up-to-date common attacks, labelled properly. The dataset is generated by following the guidelines of Gharib et al. [11] where the authors have mentioned eleven criteria to generate a reliable benchmark dataset. The data was captured over a period of 5 days, from Monday, July 3, 2017 to Friday, July 7, 2017, by profiling the abstract behavior of 25 users on a complete network topology which included firewalls, routers, modems and machines of different operating systems as well. The attacks implemented are BruteForce FTP, BruteForce SSH, DoS, HeartBleed, Web Attack, DDoS, Infiltration and Botnet. More explanation of the data collection environment and the dataset is given in [12]. We have chosen this dataset to work on as this is the most recent publicly available network data on the internet which adheres to the principles of a benchmark dataset and covers many attacks than before. The number of data instances for each type are-

- Benign –2359289
- DDoS – 41835
- PortScan – 158930
- Botnet – 1966
- Infiltration – 36
- WebAttack – 2180
- SSH – 5897
- FTP – 7938
- DoS – 252661
- Heartbleed –11

We will use this dataset for binary classification as well as multi-class classification. For binary, we use 60% of this dataset for training and 40% for validation. For multi-class classification, we use a 80-20 split. This ensures that our training set is large enough. It also ensures that our training set will validate the system with every possible type of attack it is trained to detect. By selecting the training data from the given whole dataset, we guarantee that we have labelled data, which can be used to check the accuracy of our intrusion detection system by comparing the predicted attack/benign type with the actual attack/benign type.

As we have learnt from the analysis on KDD data, U2R had lesser instances in the training set, so the classifier did not perform well. Here, there are very few data instances of Infiltration attack and Heartbleed attack. Hence, it will be interesting to see how the system handles them.

3.2 Algorithms

In [12], Iman et al. implemented 7 common machine learning algorithms on this dataset and concluded that Random Forest is the best algorithm among these as it has the lowest execution time and better values of evaluation metrics compared to the others. They have also used the RandomForestRegressor class of scikit-learn to identify which of the 80 features of the dataset helps identify each attack.

We propose to follow their example initially. We will build 2 different intrusion detection systems, 1 using Random forest and 1 using Recurrent Neural Network and compare the performance of the 2 systems.

The Random Forest algorithm works by constructing multiple decision trees based on the idea that a combination of algorithms will improve the accuracy. It grows by considering various combinations of all available features. The combinations are chosen randomly and multiple decision trees are created based on these random selections. We are using this algorithm because it has easy interpretability. We can easily see which features have been used to split the trees and which features were most important. The results are easily interpreted.

Neural Networks are known for their fast processing and accuracy. The simplest neural networks have information flowing from the input layer to the output layer via multiple hidden layers in between. Recurrent Neural Networks are different because each hidden layer receives its own output as input. This helps in comparing previous packet data with current packet data to identify discrepancies or similarities and train the system based on them. We are using this algorithm because of its success in other applications [18] as well as for harnessing its faster processing powers.

3.3 Evaluation Metrics

The validity of the built model should be checked to see if the model can be deployed in the real-world environment. If the model misclassifies most of the data of training set, then we conclude that the system is not trained properly to handle real world diverse data. If such a system is used in commercial systems, the results will not be trustworthy and may lead to catastrophic effects. To prevent this, it is important to use appropriate evaluation metrics to test the system.

The most common and easily understood metric for classification is a confusion matrix. It is a table which easily shows how many instances of data have been correctly classified and how many have been wrong.

Table 1: Confusion Matrix

		Predicted	
		Benign	Attack
Actual	Benign	TP	FN
	Attack	FP	TN

In this table, the rows depict the number of actual data instances with that class label in the training set. The columns depict the number of data instances predicted for each class label. Together they give us a complete idea of the accuracy of the system. As seen in the table, the cells, TP and TN show us the number of correctly classified instances, cell FN and FP show us the number of instances for which the system did not predict correctly.
The table is interpreted as [13]:
TP- True Positive – the number of anomalies predicted correctly.
TN – True Negative – the number of benign activities predicted correctly. From these definitions, we understand that the values of TP and TN should be high for a good classification system.
FP – False Positive – the number of anomalies predicted as benign activity. This value should be as low as possible. A high FN can be dangerous as it could harm the system. The purpose of developing an intrusion detection system is in vain if this value is high.
FN – False Negative - the number of benign activities predicted as anomaly. If the value of FP is high, it means that the system is wrongly alerting about intrusions. However, a high FP value can be tolerable; it is not harmful to the system.
These values are used in calculating other evaluation metrics as well.
Accuracy is determined by the ratio of the sum of data instances in TP and TN to the total number of data instances in the training set.
Accuracy = (TP + TN) / (TP + TN + FP + FN)

Detection Rate is the rate of anomalies detected correctly.
DR = TP / (TP + FN)
Hence, we aim to make a system with low FN value and high Accuracy and Detection Rate.
In [12], they have used other metrics, namely precision and recall.
Precision = ratio of correctly classified attacks to all predicted attacks
= TP / (TP + FP)

Recall = ratio of correctly classified attacks to all attacks
= TP / (TP + FN)
Both these values should also be high.

4 IMPLEMENTATION

4.1 Pre-processing:

The dataset had various files, each collected for a different day. The CSV files provided first had to be cleaned and transformed to suit our analysis. We took the following steps for doing so-

4.1.1 Normalization and Scaling:

We used MinMax Scaler on all the features to standardize the dataset and scale all the values between 0 and 1.

4.1.2 Handling of null values:

By doing preliminary analysis, we found out that one file had a lot of null values. There were a lot of null values in the web attacks dataset, more than the number of attacks. Dealing with this imbalance will be difficult without gather more information about the data collection scheme. Hence, we skipped this attack in our analysis. For other files, we found out that only the feature 'Flow Bytes/s' had null values and the feature 'Flow Packets/s' had infinity values. We replaced both of these values by 0.

4.1.3 Dealing with IP addresses:

The dataset contains source and destination IPs for all the flows. IP addresses are a unique numeric identification which are expressed as 4 octets, each of them separated by periods. For example, 192.168.10.16. This cannot be handled by the machine learning algorithms directly. So they have to be converted to a numeric form. We did this in 2 ways – one, we split each octet as a separate column in the dataset. The octets of the IP represent network address and host address. We split each octet separately to see the effect of these separate network and host addresses on the type of attack.

The second way to deal with IP addresses is to convert the IP notation to an integer number. Taking the same IP as above, 192.168.10.12, its integer representation is calculated as –

$256^3*192 + 256^2*168 + 256*10 + 12 = 3232238092$

All the source and destination IPs were converted to their integer representations and then processed by the algorithms. These results were compared with the results of the previous method.

4.1.4 Preparing Labels for binary classification:

The given dataset had labelled data which identified different types of attacks and benign data. For binary classification, we renamed the labels to 'attack' and 'benign' only. Also, the data was split into multiple files by day and attack, which we combined into 1 file. In doing so, we skipped the file of web attacks because of its large number of null values.

4.1.5 Dealing with Timestamps:

Timestamps are a combination of numbers and symbols like '/' and ':'. We get rid of these symbols and convert the single timestamp column into 5 columns named day, month, year, hour and minute. Each of these are then considered as a numerical variable for processing.

4.2 Models Built:

Random Forest and Recurrent Neural Networks models implemented are as follows:

- Binary Classification: Classifies whether the network flow is attack or benign.
    - For dataset with IPs as an integer
    - For dataset with IP octets split into different octets.
- Multi class classification: Classifies each type of attack in the dataset.

4.2 Random Forest:

The random forest algorithm was implemented in R using "randomForest" package. Usually there are two ways to evaluate

the model. One is to split the data into training and testing datasets. Train the model using the training data and then test if the model is correctly predicting the target variable using the test set. Second is to calculate OOB error rate. Random forest calculates OOB error rate during the training phase. We have used both the methods.

The dataset is split into training and testing dataset. The ratio of split is 60:40. To get a good detection rate, we tuned the random forest model. To do so we first identified optimum value of the following tuning parameters:

1. Number of trees (ntrees)
2. Number of features randomly sampled for splitting at each node.(m_{try})

We build the forest with different number of trees over the training dataset and also recorded the OOBerror rate for corresponding number of trees. We then plotted these results as shown in Figure 1. It can be observed that as the number of trees increases the OOB error rate decreases and the lowest error is at 30 number of trees and remains constant. Hence we have used number of trees as 30 for all the models.

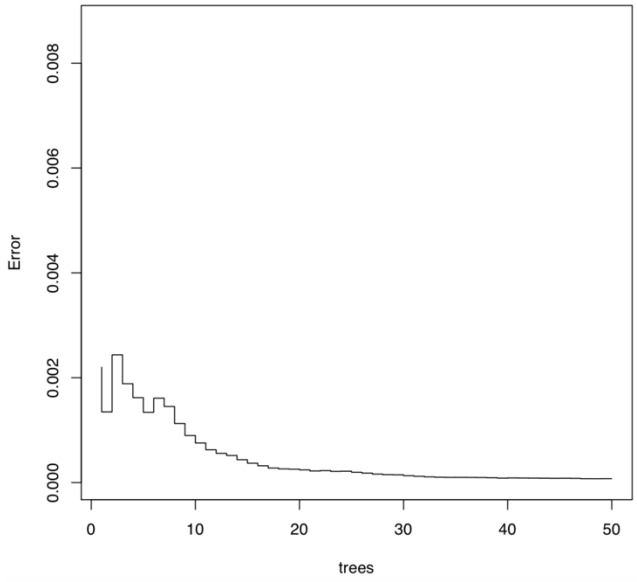

Figure 1. OOB error decreases as number of trees increases.

Similarly, we build the forest with different number of features (m_{try}) to be sampled at each node split and as shown in Figure 2, plotted it against the OOB error rate. From the figure it can be observed that OOB error rate is minimum when m_{try} is 10. Hence, we have used m_{try} as 10 for all the models.

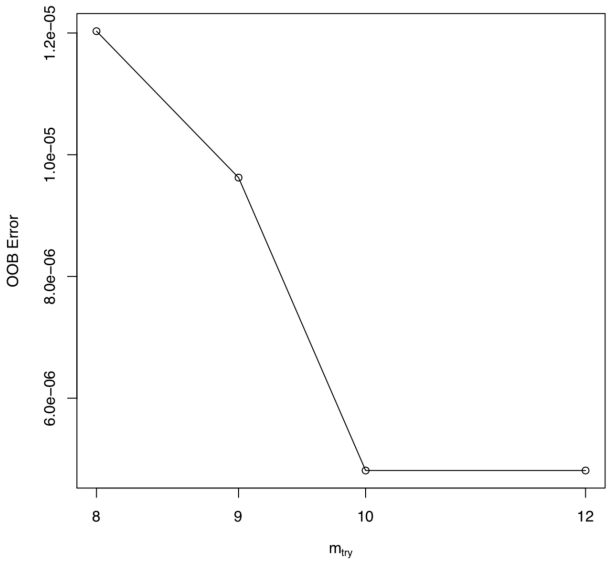

Figure 2: OOB error rates decreases as the number of features (m_{try}) decreases.

4.3 Recurrent Neural Networks

We used the keras API to implement recurrent neural networks. Keras provides easy functions to choose number of layers, activation functions, number of neurons in layers, loss functions, among other things. We used a network architecture of 1 input layer, 1 hidden layer of recurrent nodes, 1 hidden layer of normal neurons and 1 output layer. The number of neurons in each layer varied according to type of analysis.

For binary classification –

We used 60-40 split for binary classification. For the dataset with split IPs, we had 93 neurons in the input layer. For the dataset with IPs converted to integer values, we used 87 neurons in the input layer. These numbers are selected as the number of features in the cleaned and modified dataset. We used 100 neurons in the RNN hidden layer and 80 neurons in the second hidden layer. As the output was binary, we used only 1 neuron in the output layer. We used activation functions tanh and relu for the initial layers and sigmoid for the output layer. As it is a binary classification problem, 'binary crossentropy' was the appropriate loss function to use with adam optimizer. We varied the learning rate from 0.01 to 0.00000001 with decrements of 10, to see how the accuracy varies and to find the best match.

For multi-class classification –

We used 80-20 split for multi-class classification. In this case, the training set split is larger than binary classification because the dataset of individual attacks is smaller. We want to make sure there are enough number of attacks and benign data for the model to train on, so we choose a bigger split. As there are multiple attacks, the number of nodes in the output layer vary. Also, we change the activation function for the output later to softmax and change the loss function to 'categorical crossentropy'.

We then plot the loss and accuracy variations with epochs to see how the loss converges. We also produce the confusion matrix to see the number of false positives generated.

5 RESULTS

To find the best learning rate for this classification, we varied the learning rate from 0.01 to 0.00000001. These following Table 2 demonstrates the results-

Table 2. Accuracy and false positive rates with varying learning rates

Learning Rate	Accuracy (%)	False positives/94049
$1e^{-2}$	99.97	20
$1e^{-3}$	99.99	6
$1e^{-4}$	99.99	15
$1e^{-5}$	99.98	43
$1e^{-6}$	99.95	103
$1e^{-7}$	97.61	5569
$1e^{-8}$	92.58	7329

We see that we get least false positives for learning rate 0.001. The Figure 3, Figure 4, Figure 5, Figure 6 shows the plot of loss and accuracy for learning rate $1e^{-3}$, $1e^{-4}$, $1e^{-5}$, $1e^{-6}$ respectively.

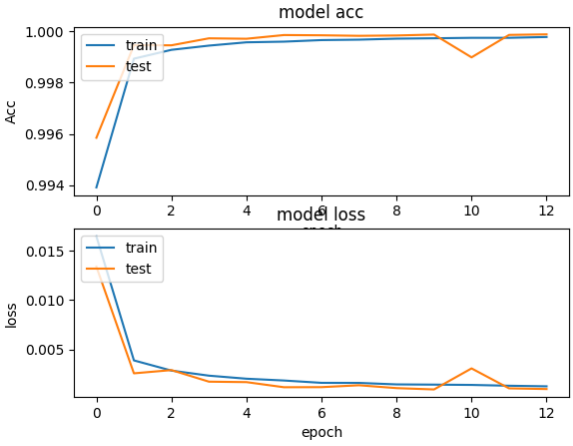

Figure 3. Loss and accuracy plots for learning rate $1e^{-3}$.

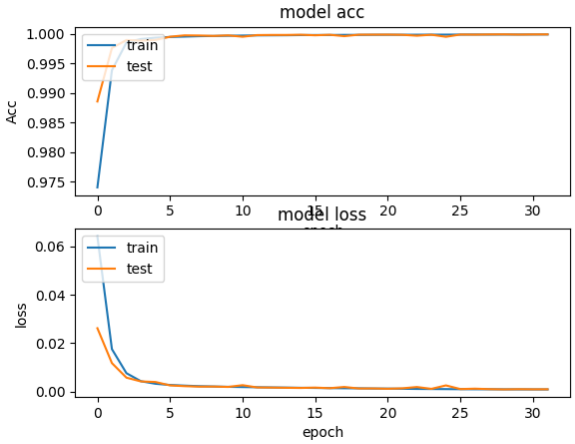

Figure 4. Loss and accuracy plots for learning rate $1e^{-4}$.

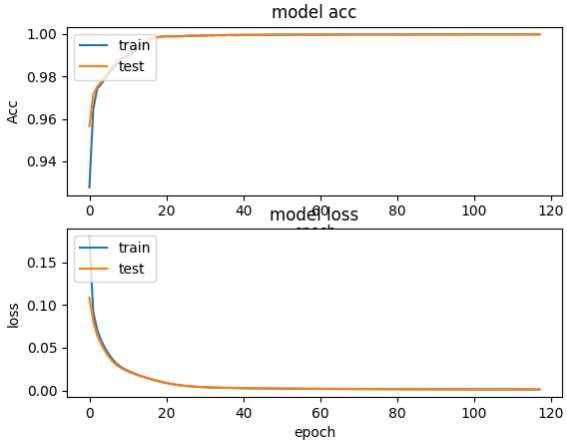

Figure 5. Loss and accuracy plots for learning rate $1e^{-5}$.

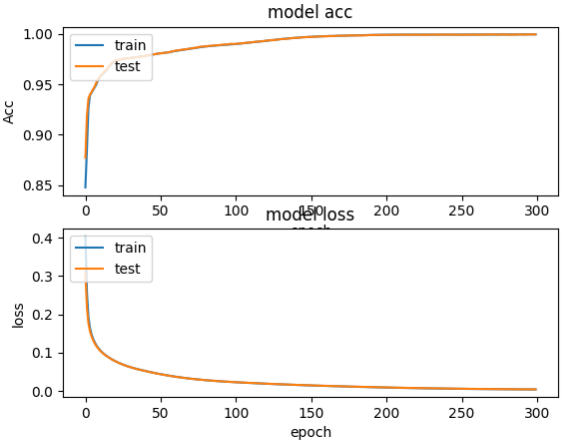

Figure 6. Loss and accuracy plots for learning rate $1e^{-6}$.

The above graphs were plotted till the number of epochs after which the difference between training loss and testing loss stops reducing. From the above figures we observe that the loss converges the

fastest for learning rate 0.001. Hence we continue the further analysis with this learning rate.

After finding the optimum parameters for both the models.: RNN and random forest, we constructed the model using training dataset which is 60% of the whole data and tested it with testing dataset which is 40% of the whole dataset.

The results for the Random forest are as follows:

The following Table 3 is a confusion matrix for dataset with IPs as an integer:

Table 3. Confusion matrix for dataset with IPs as an integer

Predicted Observed	Benign	Attack
Benign	188338	43
Attack	25	875478

Accuracy: 99.95%

The following Table 4 is a confusion matrix for dataset with IP octets split into different columns.

Table 4. Confusion matrix for dataset with IP octets split into different columns.

Predicted Observed	Benign	Attack
Benign	188355	26
Attack	22	875481

Accuracy: 99.995%

The results for the neural network are as follows:

The following Table 5 is a confusion matrix for dataset with IPs as an integer.

Table 5. Confusion matrix for dataset with IPs as an integer

Predicted Observed	Benign	Attack
Benign	187869	301
Attack	31	875950

Accuracy: 99.97%

The following Table 6 is a confusion matrix for dataset with IP octets split into different columns.

Table 6. Confusion matrix for dataset with IP octets split into different columns

Predicted Observed	Benign	Attack
Benign	187667	503
Attack	722	875259

Accuracy: 99.88%

In this we can see that Random Forest performs better than the neural networks.

In addition to binary classification, we implemented random forest for multi class classification i.e. classify types of attack. Along with accuracy, model should be evaluated based on the number of false positives as well. Because even if the accuracy of the model is good but the false positive (actual: Attack predicted: benign) rate also matters because in the real-world scenario classifying an attack as benign can be dangerous. The following Table 7 shows False positives of multi class classification

Table 7. False Positive for multi class classification of Random Forest and RNN

Type of attack	False Positive/ Total number of attack	
	Random Forest	Neural Network
DoS GoldenEye	0/1989	0/2095
DoS Hulk	0/46203	0/46154
DoS SlowHttptest	0/1100	2/1111
DoS Slowloris	0/1141	1/1160
Heartbleed	0/3	3/3
FTP Patator	0/1644	0/1551
SSH Patator	1/1221	6/1138
Infiltration	3/4	4/6
Botnet	1/410	2/372
PortScan	4/31686	1/31855
DDoS	1/8648	16/8333

Here also we can see that random forest performs better.

6 CONCLUSION

In this research work, we implemented two models for intrusion detection, Random Forest and Recurrent Neural Network. The performances of both the models was observed on the basis of their accuracy and number of false negatives. Accuracy of both the models is similar and performance of both the models is good. But the false positive numbers of random forest are better than recurrent neural network. The results of this paper will be beneficial for the research of Random Forest and Neural Network in intrusion detection systems. The results of this paper can be used for future work to use Random forest and RNN to further decrease the number of false positives to get optimum results.

REFERENCES

[1] SANS, 2001. Intrusion Detection Systems; Definition, Need And Challenges https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-systems-definition-challenges-343

[2] Ryan, J., Lin, M. J., & Miikkulainen, R. (1998). Intrusion detection with neural networks. In Advances in neural information processing systems (pp. 943-949).

[3] Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 36(1), 16-24.

[4] Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10), 11994-12000.

[5] Sangkatsanee, P., Wattanapongsakorn, N., & Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. Computer Communications, 34(18), 2227-2235.

[6] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security, 28(1-2), 18-28.

[7] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2017). A survey of deep learning-based network anomaly detection. Cluster Computing, 1-13.

[8] Mukkamala, S., Janoski, G., & Sung, A. (2002). Intrusion detection using neural networks and support vector machines. In Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on (Vol. 2, pp. 1702-1707). IEEE.

[9] Sheikhan, M., Jadidi, Z., & Farrokhi, A. (2012). Intrusion detection using reduced-size RNN based on feature grouping. Neural Computing and Applications, 21(6), 1185-1190.

[10] http://www.unb.ca/cic/datasets/ids-2017.html

[11] Gharib, A., Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2016, December). An Evaluation Framework for Intrusion Detection Dataset. In Information Science and Security (ICISS), 2016 International Conference on (pp. 1-6). IEEE.

[12] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Purtogal, January 2018

[13] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access, 5, 21954-21961

[14] Sommer, Robin, and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 2010.

[15] Sabhnani, Maheshkumar, and Gürsel Serpen. "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context." MLMTA. 2003.

[16] McHugh, John. "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory." ACM Transactions on Information and System Security (TISSEC)3.4 (2000): 262-294.

[17] Shiravi, Ali, et al. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection." computers & security 31.3 (2012): 357-374.

[18] De Mulder, W., Bethard, S., & Moens, M. F. (2015). A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, *30*(1), 61-98.

[19]