Network Intrusion Detection System Using Auto Encoder

Department of Computer Science And Engineering
Vellore Institute Of Technology , Chennai
rahul.mahajan2017@vitstudent.ac.in
putul.siddharth2017@vitstudent.ac.in

Abstract — In the recent years, the neural networks are solving a variety of problems involving complexity in engineering, science, market analysis, and finance. One of the important problems is a protection of our system against the cyber-attacks. Also the most challenging problems facing network operators today is network attacks identification due to extensive number of vulnerabilities in computer systems and creativity of attackers. Intrusion Detection System (IDS) can detect attacks by analysing the patterns of data traffic in the network. With a large amount of data that is processed in the IDS, then need to do a feature extraction to reduce the computational cost of processing raw data in IDS. Our approach uses simple as well as stacked Auto-Encoder as one of the most well-known deep learning models. The experimental results on the NSL-KDD dataset show that our approach provides substantial improvement over other deep learning-based approaches in terms of accuracy and detection rate.

Index terms: Auto encoder, stacked auto encoder, deep learning, KDD, network security, machine learning, feature extraction.

I. INTRODUCTION

An intrusion detection system (IDS) is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered. Although IDS monitor networks for malicious activity, but sometime they are also prone to false alarms. Intrusion detection systems come in different type and detect suspicious activities using different methods.
A **network intrusion detection system** (NIDS) is placed at a strategic point or points within our network, where it can monitor incoming and outgoing traffic to and from all the devices on the network. **Feature extraction** plays vital role in classification in the field of Informatics Engineering, especially in the area of network and information security. One of the challenges is the increase in volume traffic data in the growing network. As the impact of the rise in connectivity by of the Internet of Things and cloud-based services. With increasing extensive and complexity of data as well as a large number of new different attack types have an impact on system IDS to detect correct and incorrect behaviour.

IDS categorize into misuse detection and anomaly detection. A **misuse detection** technique measures its similarity between the input and the signatures of known intrusions. Thus, the known attacks can be detected immediately and reliably with a lower false-positive rate. However, **signature detection** is ineffective for detecting novel attacks. Anomaly detection techniques are designed to detect deviations from established normal usage patterns can be flagged as intrusions.

We propose to implement the automatic extraction of features on the intrusion detection system (IDS). We use Autoencoder which will perform automatic features extraction to reduce the dimensions of the data being processed. The aim is to study the various functions of the activation and the loss autoencoder hyperparameter can increase the level of accuracy of attack detection.

Thus remainder of the paper is organized as shown. Section II discusses some of the most important existing works. We have described our proposed methodology of simple and stacked auto-encoder along with other machine learning model in Section III. Section IV contains four parts. First part describes dataset, second describes data preprocessing and third talks about various performance metrices. Finally in fourth part we describe the evaluation and result. Then in Section V we gave our discussion. Further in Section VI we gave our conclusion and at the end in Section VII we gave acknowledgement.

II. EXISTING WORK

Many models have been proposed in earlier studies to overcome the limitations of the intrusion detection system. Associated with feature extraction, many research focuses particularly on decreasing feature dimensions without losing vital information.
Liu et al. implemented an anomaly detection model with the help of feature reduction with Principal Component Analysis (PCA) and classified using Neural Networks. This approach reduces to only 22 features from total of 41 features. The main advantage of this approach is the reduced computation time as the number of features decreases. However, the selection of the principal component is not optimal because it only processes certain features.
Datti and Lakhina compares the reduction feature of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) techniques. Having reduced the classification is done using the back-propagation algorithm. The result shows PCA is more accurate than LDA for smaller datasets, whereas for more massive datasets LDA is more accurate.
Bhupendra Ingre, et al. implemented an artificial neural network based IDS on the NSL-KDD dataset. The train-test dataset performed in 41 and selected only 29 features NSL-KDD dataset with various values of neural network architecture and the results comparing in binary and five classes. In the binary classification with 29 features, the accuracy is obtained at 81.2%.
Quamar, et al. have proposed a deep learning based self-taught learning (STL) to build effective and flexible NIDS using the NSL-KDD datasets. It mainly uses deep learning for dimensional reduction and then uses softmax regression for classification.
Zhao et al. implemented a survey of deep learning applications within machine health monitoring. They experimentally compared

conventional machine learning methods against four common deep learning methods (auto-encoders, Restricted Boltzmann Machine (RBM), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). Their work concluded that deep learning methods offer better accuracy than conventional methods.

III. PROPOSED METHODOLOGY

In our model, we are implementing algorithm for Network Intrusion Detection System (NIDS) in which we have taken NSL-KDD dataset regarding various malicious attack which infect our network. We have implemented simple auto-encoder as well as stacked auto-encoder and compared their accuracy. Also we have implemented machine learning algorithms like random forest and logistic regression to compare their results. First we have imported various dependencies which are required for our implementation. After that we have imported the NSL-KDD dataset from online github repository. Training dataset from https://raw.githubusercontent.com/defcom17/NSL_KDD/master/KDDTrain%2B.csv and testing dataset from https://raw.githubusercontent.com/defcom17/NSL_KDD/master/KDDTest%2B.csv. After analysis of the dataset we came to know that we have 148517 rows and 42 columns. Also the training and testing dataset differ by some amount of different type of attacks.
Now we start data pre-processing by making 5 different class as -
dos_attacks=["snmpgetattack","back","land","neptune", "smurf","teardrop","pod","apache2","udpstorm", "processtable","mailbomb"]
r2l_attacks=["snmpguess","worm","httptunnel","named", "xlock","xsnoop","sendmail","ftp_write","guess_passwd", "imap","multihop","phf","spy","warezclient","warezmaster"]
u2r_attacks=["sqlattack","buffer_overflow","loadmodule", "perl","rootkit","xterm","ps"]
probe_attacks=["ipsweep","nmap","portsweep","satan", "saint","mscan"] based on facts. Now categorize those various malicious attack based on the facts to these classes. Now we have 3 such rows which have categorical data, therefore we have used label and one-hot encoding which increases the number of columns to 122. Finally we have used standard scalar to transform our data such that its distribution will have a mean value 0 and standard deviation of 1.
Now we use our splitted data for training and testing purpose to apply **Random Forest Classifier** and computing its accuracy which comes out to be 0.7587. Also we applied **Logistic Regression** to our Splitted dataset and obtained accuracy of 0.7420.
Further we have implemented **stacked auto-encoder** to analyze accuracy of our system. Stacked auto-encoder is a neural network consisting of multiple layers of sparse auto-encoders in which the outputs of each layer is wired to the inputs of the successive layer. After dimensionality reduction with the help of stacked auto-encoder our number of columns come down to 10 and after this we apply Random Forest Classifier to get accuracy of 0.71. We have tested our data and found that there are 14595 Normal, 5997 DoS, 1892 Probe and 59 R2L attacks are there.
Also we have evaluated our results based on various performances metrics such as accuracy, recall, precision, f1 score and detection rate for each of the possible labels.

IV. EXPERIMENTAL STUDY

NSL-KDD Dataset –
We have used NSL-KDD dataset which is a benchmark for ML based intrusion detection, however , it suffers from several inefficiencies such as class imbalance, where for instance in the NSL-KDD training dataset only 0.04% of the samples belong to the u2r attack type making it severely underrepresented, the case is similar for the r2l and probe attack types whereas the majority of attack records are representing the DDOS attack type, this fact made it difficult for classifiers to detect these underrepresented types resulting in poor accuracy. Another issue is that this dataset is unrealistic, in reality most traffic in a network is benign and only a small percentage might be malicious, while in the NSL-KDD training set for example, attack samples compose 80% of the entire dataset which makes the models trained using this dataset ineffective in real life situations. Out autoencoder based approach attempts to overcome these problems.

Data Preprocessing –
According to the input of neural network, input vector have to be numerical value, therefore, we convert nominal features into numerical value using one-hot encoding. In the dataset, “protocol_type”, “service”, and “flag” features has nominal value. We applied the encoding on these three features, hence our feature dimension map transformed into 122 features.

The rest of 38 features have numerical value and some features have a large difference between maximum and minimum values. Therefore, we performed a Min-Max Normalization to obtain the identical value.
In the last step, both datasets (training and testing) are normalized, then converted to the desired data type.

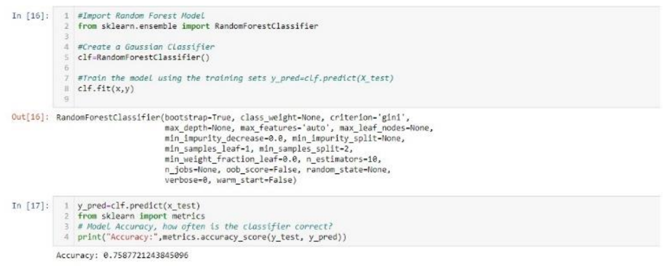
Performance Evaluation –
(i) Accuracy (AC): The ratio of correctly predicted connections over the total network traffic.
$AC = (TP + TN) / (TP + TN + FP + FN)$

(ii) Precision (P): It (also called positive predictive value) is the fraction of relevant instances among the retrieved instances.
$P = (TP) / (TP + FP)$

(iii) Recall (R): It (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved.
$R = (TP) / (TP + FN)$

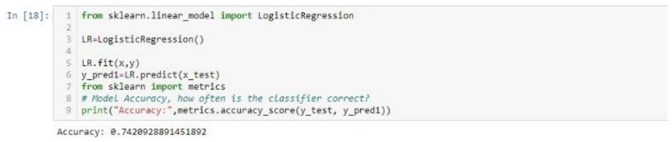
(iv) F1 Score (F1): It conveys the balance between the precision and the recall.
$F1 = 2*((P*R) / (P+R))$.

Experiment Result –
Initially we obtained accuracy before dimensionality reduction using basic machine learning algorithm such as –
1. RandomForestClassifier (using autoencoder)



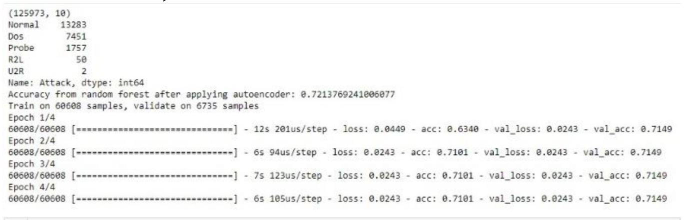
```
In [16]:  #Import Random Forest Model
          from sklearn.ensemble import RandomForestClassifier

          #Create a Gaussian Classifier
          clf=RandomForestClassifier()

          #Train the model using the training sets y_pred=clf.predict(X_test)
          clf.fit(x,y)

Out[16]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=10,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)

In [17]:  y_pred=clf.predict(x_test)
          from sklearn import metrics
          # Model Accuracy, how often is the classifier correct?
          print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

          Accuracy: 0.7587721243845096
```

2. Logistic Regression (using autoencoder)



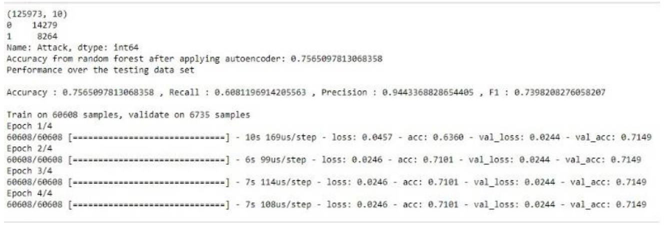
```
In [18]:  from sklearn.linear_model import LogisticRegression

          LR=LogisticRegression()

          LR.fit(x,y)
          y_pred1=LR.predict(x_test)
          from sklearn import metrics
          # Model Accuracy, how often is the classifier correct?
          print("Accuracy:",metrics.accuracy_score(y_test, y_pred1))

          Accuracy: 0.7420928891451892
```

After doing dimensionality reduction using stacked autoencoder (reducing column size from 122 to 10) we again use same machine learning (RandomForestClassifier) algorithm to calculate accuracy and it comes out to be 0.72(approx.) which is lesser than the accuracy which we calculated before doing dimensionality reduction. The accuracy of stacked auto-encoder for overall model came out to be 0.7149. Out of the testing dataset we found that Normal attack are 13283, Dos are 7451, Probe are 1754, R2L are 50 and U2R are 2.



```
(125973, 10)
Normal    13283
Dos        7451
Probe      1757
R2L          50
U2R           2
Name: Attack, dtype: int64
Accuracy from random forest after applying autoencoder: 0.7213769241006077
Train on 60608 samples, validate on 6735 samples
Epoch 1/4
60608/60608 [==============================] - 12s 201us/step - loss: 0.0449 - acc: 0.6340 - val_loss: 0.0243 - val_acc: 0.7149
Epoch 2/4
60608/60608 [==============================] - 6s 94us/step - loss: 0.0243 - acc: 0.7101 - val_loss: 0.0243 - val_acc: 0.7149
Epoch 3/4
60608/60608 [==============================] - 7s 123us/step - loss: 0.0243 - acc: 0.7101 - val_loss: 0.0243 - val_acc: 0.7149
Epoch 4/4
60608/60608 [==============================] - 6s 105us/step - loss: 0.0243 - acc: 0.7101 - val_loss: 0.0243 - val_acc: 0.7149
```

Now we have classified attack as 0 and 1 for non-attack and attack respectively. Also we applied the same logic as we done before. We got result non-attack as 14279 and attack as 8264. This time accuracy came out to be 0.7565 using RandomForestClassifier. Also accuracy is 0.6081, precision is 0.9443 and f1 score is 0.7398.



```
(125973, 10)
0    14279
1     8264
Name: Attack, dtype: int64
Accuracy from random forest after applying autoencoder: 0.7565097813068358
Performance over the testing data set

Accuracy : 0.7565097813068358 , Recall : 0.6081196914205563 , Precision : 0.9443368828654405 , F1 : 0.7398208276058207

Train on 60608 samples, validate on 6735 samples
Epoch 1/4
60608/60608 [==============================] - 10s 169us/step - loss: 0.0457 - acc: 0.6360 - val_loss: 0.0244 - val_acc: 0.7149
Epoch 2/4
60608/60608 [==============================] - 6s 99us/step - loss: 0.0246 - acc: 0.7101 - val_loss: 0.0244 - val_acc: 0.7149
Epoch 3/4
60608/60608 [==============================] - 7s 114us/step - loss: 0.0246 - acc: 0.7101 - val_loss: 0.0244 - val_acc: 0.7149
Epoch 4/4
60608/60608 [==============================] - 7s 108us/step - loss: 0.0246 - acc: 0.7101 - val_loss: 0.0244 - val_acc: 0.7149
```

V. DISCUSSION

There are two approaches applied for the evaluation of NIDSs. The majorly used approach, the training data is used for both training and testing either using n-fold cross-validation or splitting the training data into training, cross-validation, and test sets. NIDSs based on this approach achieved very high-accuracy and less false-alarm rates. On the other hand, the second approach uses the training and test data separately for the training and testing. Since the training and test data were collected in different environments, the accuracy obtained using the second approach is not as high as in the first approach. So, we give importance on the results for the second approach in our work for the accurate evaluation of NIDS.

VI. CONCLUSION

In this paper, we have discussed the problems faced by existing NIDS techniques. In response to this we have proposed our novel method for unsupervised feature learning. We have then built upon this by proposing a novel classification model constructed from stacked deep auto encoder and the Random forest classification algorithm. Our results have showed that our approach offers high level of accuracy, precision and recall all together with reduced training time. Most notably, we have compared our stacked deep auto encoder model against the machine learning classification algorithm unlike most previous work, we have evaluated the capabilities of our model based on the benchmark dataset revealing a consistent level of classification accuracy.

VII. ACKNOWLEDGMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Dr. (Prof.) Syed Ibrahim., Associate Professor, School of Computer Science Engineering, for his consistent encouragement and valuable guidance offered to us from time to time in order to rectify our errors and bring professionalism in our projects.

Putul Siddharth

Rahul Mahajan

REFERENCE

[1]

https://www.covert.io/research-papers/deep-learning-security/A%20Deep%20Learning%20Approach%20for%20Network%20Intrusion%20Detection%20System.pdf

[2]
https://www.researchgate.net/publication/329800726_Automatic_Features_Extraction_Using_Autoencoder_in_Intrusion_Detection_System

[3]

https://blogs.qub.ac.uk/netsec/wp-content/uploads/sites/153/2019/02/08264962.pdf