Application of Machine Learning in Network Intrusion Detection System (NIDS)

Yang Zhang[1], Zunayed Mahmud[2], Prithila Angkan[2]

Abstract—With the massive usages of the internet, public network security research becomes more crucial to detect the unauthorized intrusions on network traffic. In this project, we used three machine learning algorithms that are Deep Neural Network (DNN), Support Vector Machine (SVM) and Naive Bayes (NB) model on two popular network intrusion datasets. As shown in the result section, the deep neural network model outperforms the other two techniques on both KDDcup 99 and UNSW-NB15 dataset with the accuracies of 99.8% and 85.07% respectively. Finally, the performance of each model was described explicitly and compared based on the result received from the corresponding confusion matrix.

Keyword—deep neural network, support vector machine, Naive Bayes, KDDcup 99, UNSW-NB15

- *1. School of Computing, Queen's University*
- *2. Electrical and Computer Engineering, Queen's University*

1 Introduction

In 2019, IBM Security reports an annual cost of Data Breach that is collected from 507 organizations in 16 countries, the global average cost of a data breach in 2019 is $ 3.92 million. On average, the U.S. loses $ 8.19 million due to the data breaches. More than half of the data breaches are caused by malicious attacks or intrusion [1], which makes the development of intrusion detection systems (IDS) more urgent and significant. The intrusion detection system is a software application that monitors and collects security information from network traffic or operation systems for detecting malicious behaviors from unauthorized users [2]. Intrusion detection systems can be categorized into three types in terms of target systems which are: network-based intrusion detection systems (NIDS), host-based intrusion detection systems (HIDS) and hybrid. From detection criteria, it is grouped into three types as well: signature-based detection, anomaly-based detection and hybrid [3]. Datasets applied in this project are benchmark dataset KDDcup99 [4] and the most recent and popular dataset UNSW-NB15 [5] composed by network traffic data. In network intrusion detection systems, the objective is to detect malicious attacks from the key nodes in the entire network traffic, and the identification processes are usually performed by machine learning methods. In an example of an artificial neural network-based NIDS [6], large volumes of data are handled efficiently in terms of computational time and power, which also persists a high detection and prediction rate. The first statistics model was proposed for IDS in 1986 by Dorothy E. Denning and Peter G. Neumann, which formed the basis of many current systems [7]. Machine learning techniques have always played an important role in the intrusion detection system model development. Nowadays, with the rapid development of machine learning, it substantially boosts the progress to construct novel IDS models. Although there has been plenty of research performed and innovative approaches outlined with temptations to solve the intrusion detection problem, more challenges are continuously found in this area. For instance, how to deal with the enormous amount of increasing information from a large network? How to build up a system that can monitor and collect real-time data? How to leverage the existing techniques to help with IDS and Why?

In this paper, we have applied three machine learning models on two splendid datasets that are KDDcup99 and UNSW-NB15. Later we have evaluated our model performances from the following three aspects:

1. Comparing the performance of each model in KDDcup99 dataset
2. Comparing the performance of each model in UNSW NB 15 dataset
3. Describe the performance comparison of each model between two datasets

Machine learning approaches employed here are deep neural networks (DNN), support vector machines (SVM) and Naive Bayes Model (NBM). Compared with the other traditional classification methods, such as regression model, knn or the Naive Bayes model, DNN is expected to have usually higher accuracy and prediction rate with a low false-positive rate under the scenario of big data size [11]. This can be also explained by the principle of DNN that extracts high dimensional features locally in the Euclidean space, which denoises backgrounds or highlights the features from many small neighbors in the intrusion network [12]. For performance comparison with the deep learning model DNN, we have selected SVM and NB as our reference. SVM is a traditional rule-based machine learning method, which works well especially on binary datasets, such as UNSW-NB15. Naive Bayes model is presented as a statistical model to show a comparison with the other two different types of machine learning models. We have successfully applied all these three algorithms on both the dataset and received the highest accuracy using the DNN model which is 99.8% and 84.36% respectively. The rest of the paper is described in the following sections: dataset description, methodology, evaluation metric, result discussion, conclusion.

2 Dataset Description

KDDcup99:

KDDcup99 dataset is the most commonly used dataset for anomaly-based Network Intrusion Detection System (NIDS). It was generated from an evaluation program of IDS known as the DARPA'98 [13]. The MIT Lincoln Labs prepared DARPA'98 in the year 1998 with the intention to contribute to the research related to IDS [14]. DARPA consists of 4 gigabytes of compressed binary TCP dump data which are collected over a period of 7 weeks of network traffic. The data is then processed into a 5 million connection record of 100 bytes each [14]. The test data has about 2 million connection records [14]. KDDcup99 has 4,898,431 data points in training dataset and 2,984,154 data points in test dataset also there are 41 features that fall under one of the 3 categories which are Basic, Traffic and Content [15]. There are 4 attack classes in KDDcup99 which are DoS, U2R, R2L and Probing Attack along with one Normal class [15].

One of the main problems of this dataset is the duplicate data where 78% training data and 75% test data are duplicate [16]. This duplication causes the dataset to be skewed, as a result the machine learning algorithms are more intended to learn from the majority attacks and detect them whereas the model will be susceptible towards the minority attacks such as U2R or R2L [15]. Apart from the data duplication, the other problem in KDDcup99 dataset is the feature redundancy. In the paper [15], they have found out 17 irrelevant features using Mean Decrease Impurity metric.

UNSW- NB15:

This dataset contains the up to date normal activities and synthetic contemporary attack behaviours and was generated in the year 2015 in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) using IXIA PerfectStorm tool3 to create a hybrid of the modern normal and abnormal network traffic. [5]. Originally it had 49 features with the class label, however the revised version of the dataset contains 41 features. In our project we will be working with the revised version of the dataset which has no missing values and contains 175,341 training data points and 82,332 test data points [15]. There are 9 attack classes which are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell Code and Worms along with one Normal class [15]. This dataset is less skewed and the distribution between the training and test set is stationary [15].

2.1 Data Preprocessing

We firstly downloaded 10% KDDcup99 training data and 10% KDDcup99 test data, where the training set contains 494020 records and 42 features displaying on Figure 1. In the test set, it includes 311026 records and 42 features visualizing on Figure 2. In order to fit in our machine learning models with the unbiased, denoised, informative inputs, the following preprocessing steps have been proposed:

1. Investigating the dataset, and removing missing values and wrong format records.
2. Converting data from symbolic to numerical by giving the values based on their positions in each data category (feature).
3. Normalizing and transforming the numerical data into [0, 1] by considering the range of each feature.

As demonstrated in Figure 1 and 2, both 10% training and test data have no missing values, however, some records contain the faulty format in the dataset, such as the protocol feature variables have been put in services feature column, all those records have been removed entirely. From the KDDcup99 website, we download the annotation file to convert symbolic data to numerical. There are 4 symbolic features/categories that need to be preprocessed, such as protocol, service, flag and attacks types. In those symbolic features, we have 3 types in protocol, 70 types in service, 11 types in flag and 40 types of attacks. We convert protocol, service and flag features into numerical values based on its position for different types from the annotation file. For instance, "icmp" is on the third place in protocol feature, so it would be converted as number 2. The attack feature is converted based on the subcategory of attacks, for example, "back, smurf" belongs to denial-of-service attack/dos attack, while dos attack is on the second place in attack feature from annotation file, so all the dos attacks would be converted into number 1. Moreover, in the test dataset, there are 17 more attack types than training dataset. To keep consistency as in the training set, we removed all those records. A normalization method has been used as suggested in the paper [12]. It transforms all numerical data into range [0, 1] by applying the following equation on each feature:

$$\hat{X} = \frac{x_i - X_{max}}{X_{max} - X_{min}}.$$

Where the feature variable $X = [x_1, x_2, \dots, x_m]$ is normalized as $\hat{X}$, the maximum value $X_{max} = max(X)$ and $X_{min} = min(X)$. The m is the numbers of records/rows. Finally, the one-hot encoding method is performed to have the same weights and to avoid the bias for the classifier models later. After the preprocessing step, the final datasets include 494020 records for training and 292297 records for testing.

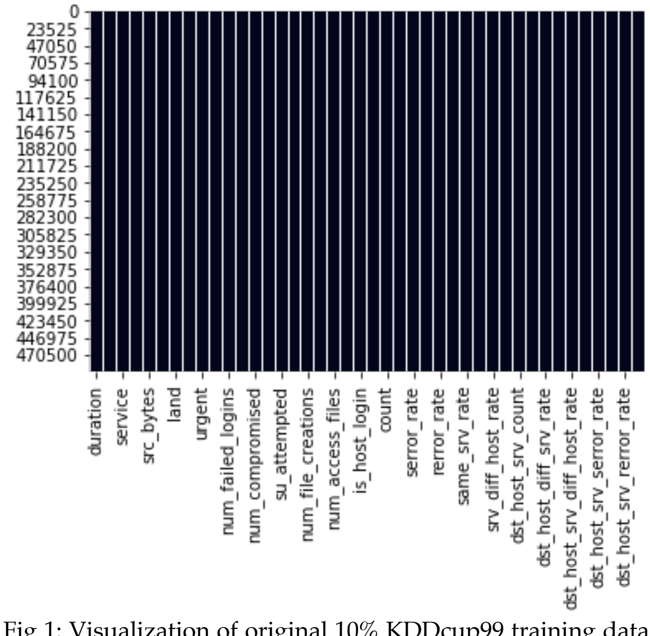


Fig 1: Visualization of original 10% KDDcup99 training dataset for the detection of missing values, which includes 494020 records and 42 features. The black bars show the valid inputs, while no presentation of white gaps displaying as missing or invalid variables. Missing value detection is processed by applying python packages "seaborn" and "missingno".

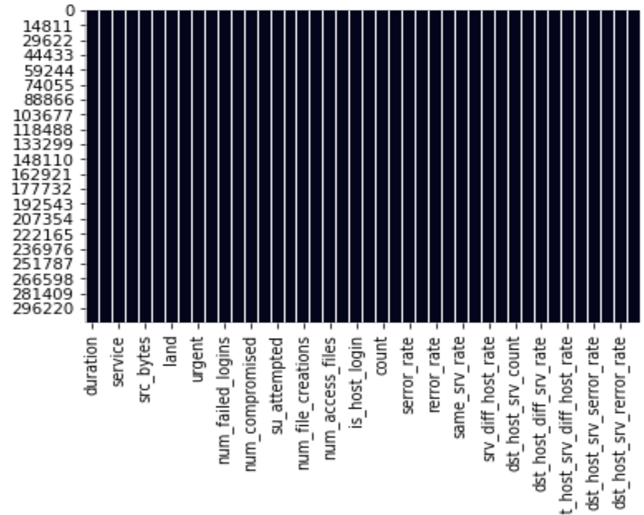


Fig 2: Visualization of original 10% KDDcup99 test dataset for the detection of missing values, which includes 311026 records and 42 features. The black bars show the valid inputs, while no presentation of white gaps displaying as missing or invalid variables. Missing value detection is processed by applying python packages "seaborn" and "missingno".

For UNSW-NB15, there are no missing values which has been shown in Fig 3 where the black bars show that there is no missing value in both train and test set. However there are several columns with categorical values that we needed to convert into numerical values in order to fit the models. While exploring the dataset we found that there is a significant imbalance in the 'proto' and 'state' attribute where the train set had two unique values in the 'proto' attribute which are 'icmp' and 'rtp' and the test set had two unique values in the 'state' attribute which are 'ACC' and 'CLO'. That means, if we apply the encoding technique in each individual training and test dataset , it would lead to a difference in dimensionality. Therefore, we found that it would be convenient if we concatenate the training and test dataset and then apply the encoding rather than doing them individually in each dataset. Afterwards, we splitted the concatenated dataset for training and testingOnce the merging was done, we applied one hot encoding and label encoding to convert values from nominal to numerical. The original dimension of the training and test dataset was increased and both the dataset were later normalized using min-max normalization that we did for the KDDcup99 data. After that, the data was splitted into a training and testing set where 70 percent of the data was used as a train set and the remaining was used as a test set. After this split, the new dimension has become 180371 x 207 in the training set and 77302 x 207 in the test set.

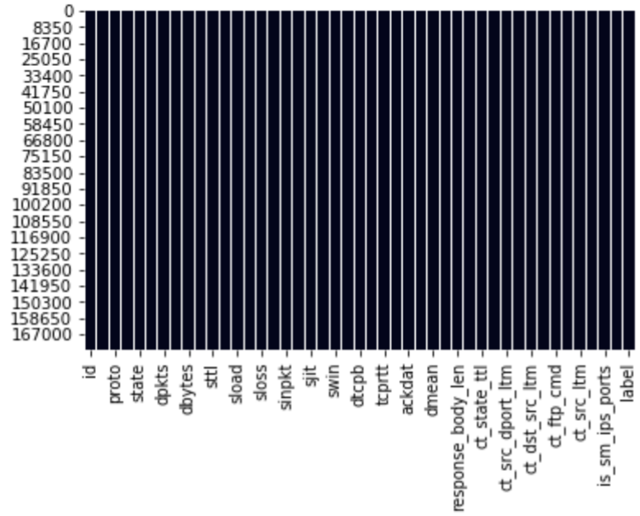


Fig 3: Visualization to detect missing values in UNSW-NB15 original training set where the black bars show the valid inputs, while no presentation of white gaps displaying as missing or invalid variables

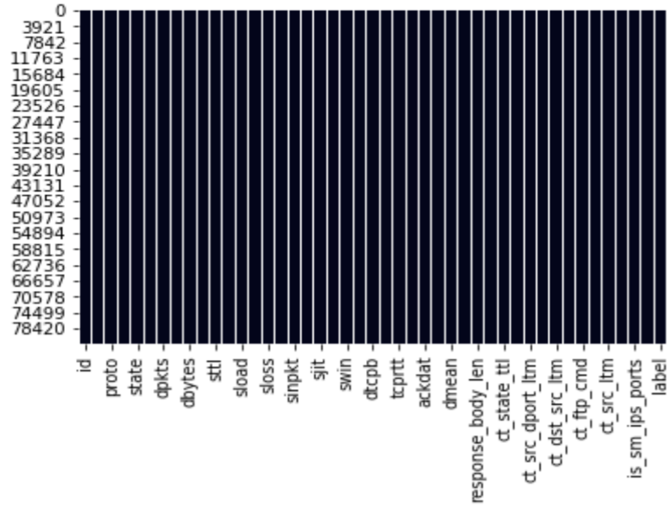


Fig 4: Visualization to detect missing values in UNSW-NB15 original test set where the black bars show the valid inputs, while no presentation of white gaps displaying as missing or invalid variables

The target distribution graph shows the distribution of attack and normal class within the dataset which helps to understand if the dataset is skewed or not. In the UNSW- NB15 dataset we can see from Figure 7 that around 68 % of the training data belong to the attack category whereas in the test set it is around 55 %. More data belonging to the attack class of the training test allows the machine learning model to get trained to detect the attacks more which is only key factor to building machine learning models for intrusion detection.

In order to train our machine learning models properly, we investigate the distributions of target feature/attack in both KDDcup99 and UNSW-NB15. As shown in Figure 5-8, KDDcup99 is more skewed than UNSW-NB15 as described. More than half of the attacks in KDD are in the "DoS" attack category in training and test sets, which might have influences on the performances. For the KDD dataset, multi-category classification is kept and analyzed throughout the project, while the classes in the UNSW-NB15 dataset have been combined into a binary variable as the number of non-normal attack types are

too low and create noise for data analyses. The aforementioned categories in the KDDcup99 dataset and UNSW-NB15 dataset are re-labeled numerically (see the lists below) and applied in the entire modeling processes, model evaluations and consequential conclusion part.
Category label for the KDDcup99 dataset:

- ❑ The normal category labeled as '0';
- ❑ The DoS attack category labeled as '1';
- ❑ The Probing attack category labeled as '2';
- ❑ The R2L attack category labeled as '3';
- ❑ The U2R attack category labeled as '4'.

Category label for the UNSW-NB15 dataset:

- ❑ The normal category labeled as '0';
- ❑ The anomaly attack category labeled as '1'

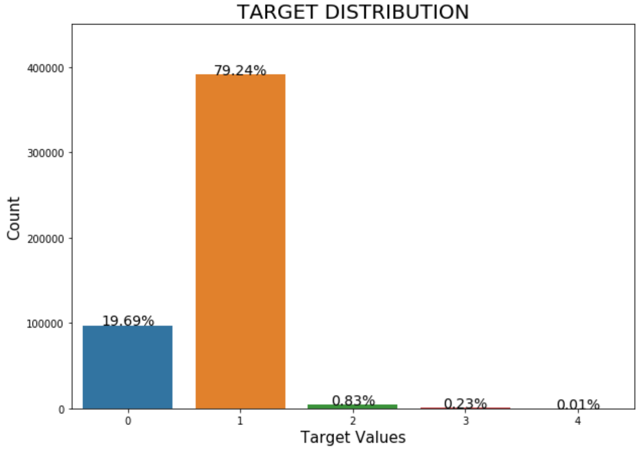


Fig 5: A visualization of 5 different target distributions of the training dataset in 10% KDDcup99. The normal class (0), the Dos attack (1), the Probing attack (2), the R2L attack (3) and the U2R attacks (4).

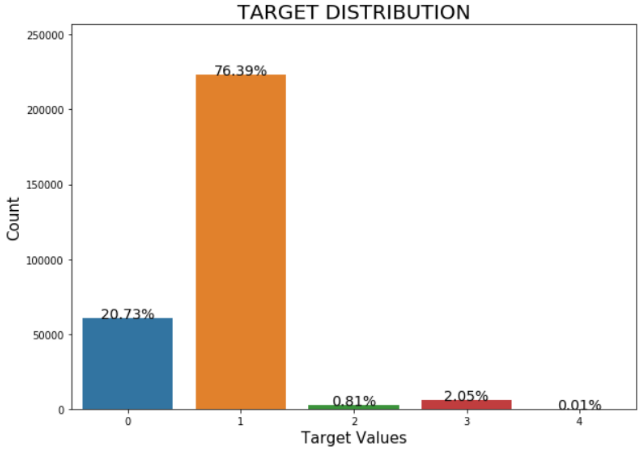


Fig 6: A visualization of 5 different Target distributions of the test dataset in 10% KDDcup99. The normal class (0), the Dos attack (1), the Probing attack (2), the R2L attack (3) and the U2R attacks (4).

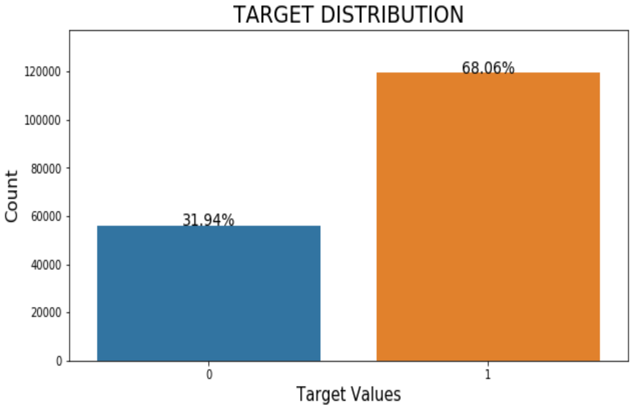


Fig 7: Target distribution of the training dataset where 0 is normal and 1 is attack class

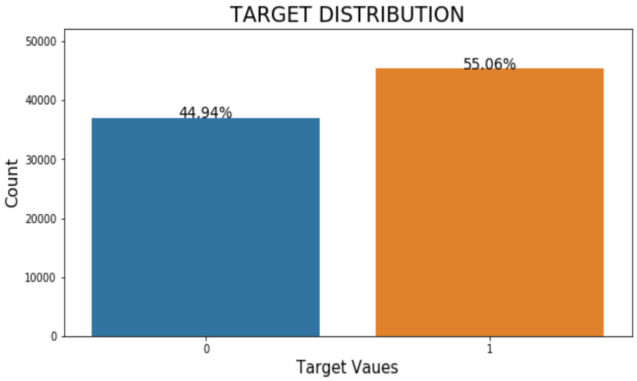
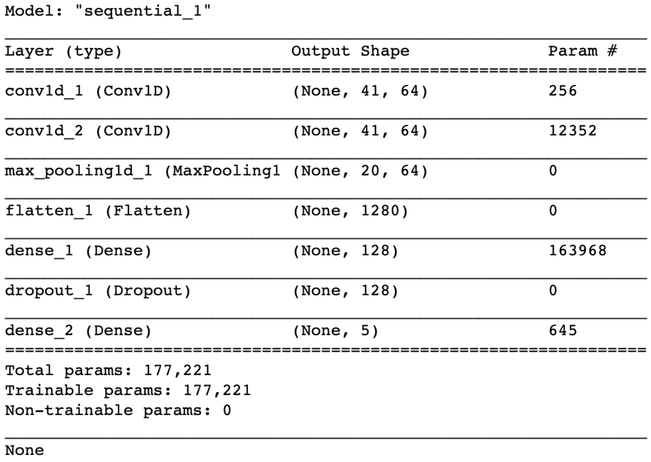


Fig 8: Target distribution of the test dataset where 0 is normal and 1 is attack class

3 METHODOLOGY

3.1 Deep neural network

We propose a shallow neural network model with less convolutional layers. Max Pooling method is used to reduce the dimensionality of the features maps, and Flatten function is performed to connect neurons fully. Then, we use the Dense_1 activation function to reduce the dimensions of the weighted neurons from the previous fully connected layer to 128 outputs. In order to prevent overfitting, we set up a dropout rate at 0.5, and finally, we use the Dense_2 activation to keep reducing dimensions from 128 to the final 5 different attack types as prediction outputs. The entire process constructs 177, 211 parameters, which the architecture is displayed in Figure 9.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_1 (Conv1D)            (None, 41, 64)            256       
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 41, 64)            12352     
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 20, 64)            0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1280)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               163968    
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 645       
=================================================================
Total params: 177,221
Trainable params: 177,221
Non-trainable params: 0
_________________________________________________________________
None
```

Fig 9: The architecture of Convolutional Deep Neural Network model, which contains 2 layers of convolution, 1 layer of max pooling, 1 fully connected layer and two dense layers.

3.2 Support Vector Machine & Naive Bayes Classifier

Support Vector Machine (SVM) is a supervised machine learning method that performs classification on labeled data. In our project we are using the basic SVM model with RBF (Radial Basis Function) kernel and C= 0.5. We imported SVC (Support Vector Classifier) from the scikit-learn library. The reason we used RBF kernel is that it provides better accuracy than linear kernel. We tried to keep the value of the regularizer C to the commonly used 0.5 to keep the model simple yet not allowing much misclassification of the data points.

Naive Bayes is a simplified probabilistic classifier that makes naive assumptions about the features in a dataset to be independent of each other [18]. We imported the GaussianNB classifier from the scikit-learn library. The reason we are using Naive Bayes is that it is a faster and very powerful algorithm[19]. Other than that, we would like to leverage the unbiasedness of Naive Bayes to overcome the skewness issue in our data and make the model balanced over different classes or features because Naive Bayes considers each class and feature independently.

4 EVALUATION METRIC

In order to evaluate the performance of each algorithm that has been applied on both the dataset, we have used the classification accuracy as the primary evaluation metric and later we ran the confusion matrix for all the algorithms to check how many attacks were correctly identified. The classification is multiclass for the KDDcup99 dataset and binary for the UNSW-NB15 dataset. For the KDDcup99 dataset, the confusion matrix is a 5x5 matrix containing four different categories of attacks and a normal category. On the contrary, the confusion matrix of the UNSW-NB15 dataset is a 2x2 matrix since all the attacks in this dataset are labeled as ‘1’ and the normal records are labeled as ‘0’. The reason we are using confusion matrices as one of the evaluation criteria is that it explains how many of the attacks the models have clearly identified along with the normal records as opposed to the classification accuracy which provides the overall accuracy.

5 RESULTS DISCUSSION

As mentioned in the previous section, the performance of the algorithms has been evaluated based on the classification accuracy. Fig10 shows the accuracies that we have received from the two dataset using the three algorithms. As shown in the graph, we have received the highest accuracy in both the dataset using the deep neural network model. The best accuracies achieved using this algorithm are 99.8% on KDDcup99 dataset and 85.07% on UNSW-NB15 dataset which is significantly higher compared to the other machine learning algorithms that we have applied.

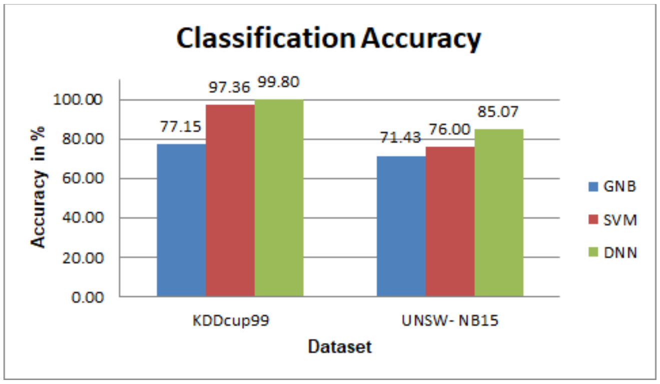


Fig 10: Performance comparison by classification accuracy

In order to have more informed information as to where the algorithms have failed and how many attacks they were able to classify and how many they did not we have created the confusion matrix for each of these algorithms on both the dataset.

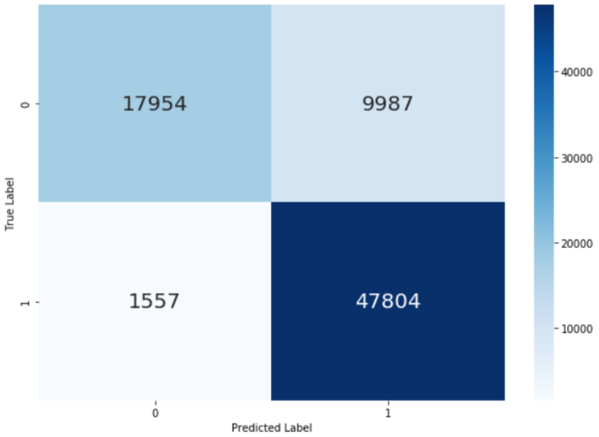
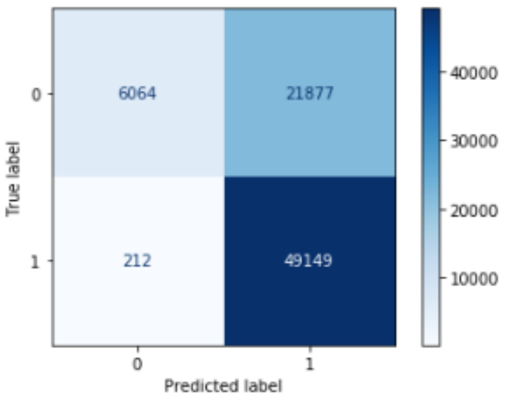


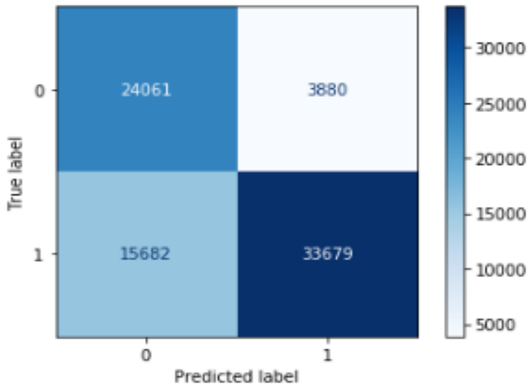
Fig 11: Confusion matrix on UNSW-NB15 using DNN

The 2x2 confusion matrix in Fig11, shows that the DNN model successfully identified 47,804 attacks out of 49,361 attacks and 17,954 normal records out of 27,941 normal records. Similarly, Fig12 and Fig13 shows the confusion matrix on NB and SVM classifier where the models have identified 49,149 attacks out of 49,361 along with 6064 normal records out of 27,941 and 33,679 attacks out of 49,361 along with 24,061 normal records out of 27,941 respectively.



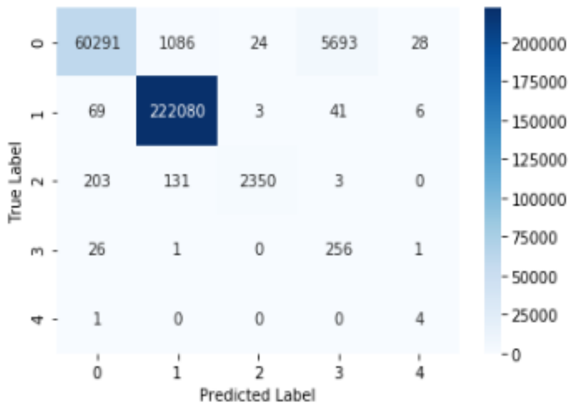
	0	1
0	6064	21877
1	212	49149

Fig 12: Confusion matrix on UNSW-NB15 using NB



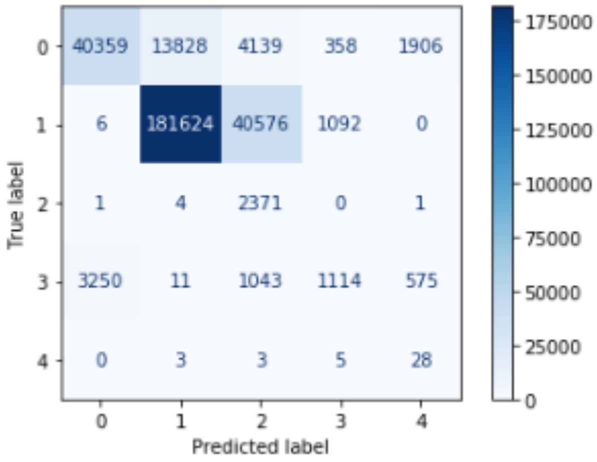
	0	1
0	24061	3880
1	15682	33679

Fig13: Confusion matrix on UNSW-NB15 using SVM



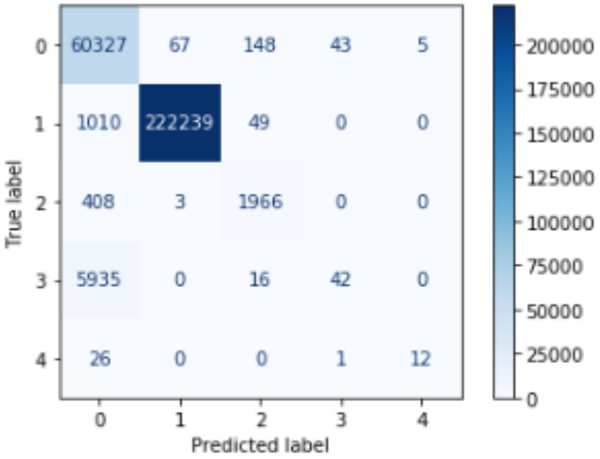
	0	1	2	3	4
0	60291	1086	24	5693	28
1	69	222080	3	41	6
2	203	131	2350	3	0
3	26	1	0	256	1
4	1	0	0	0	4

Fig 14: Confusion matrix on KDDcup99 using DNN



	0	1	2	3	4
0	40359	13828	4139	358	1906
1	6	181624	40576	1092	0
2	1	4	2371	0	1
3	3250	11	1043	1114	575
4	0	3	3	5	28

Fig 15: Confusion matrix on KDDcup99 using NB



	0	1	2	3	4
0	60327	67	148	43	5
1	1010	222239	49	0	0
2	408	3	1966	0	0
3	5935	0	16	42	0
4	26	0	0	1	12

Fig 16: Confusion matrix on KDDcup99 using SVM

The 5x5 confusion matrix portrayed in Fig14 shows that the DNN model has successfully identified 2,22,080 DoS attacks out of 2,23,298 which is the highest attack instance in this dataset. Fig15 and Fig16 show the confusion matrix using NB and SVM algorithms respectively. It can be seen that although the algorithms failed to correctly identify most of the normal records and the attacks that have very low occurrence in this dataset compared to DoS attack, the SVM algorithm has shown better performance identifying 2,22,239 DoS attacks out of 2,23,298 compared to NB which identified only 1,81,624. However, in terms of the comparison from all the confusion matrix on both the dataset and the classification accuracy it can be seen that the DNN model outperforms the other two algorithms. However, the reason we are getting different accuracies even by using the DNN model is the nature of the two different dataset. According to the general convention, neural networks require larger numbers of data to train and get a good classification accuracy which explains the reason for getting higher accuracy in KDDcup99 data as opposed to the UNSW-NB15 data that has lower instances. Moreover, the dimensionality of the data after preprocessing is a factor against the accuracy that we received for all the algorithms. After preprocessing the dimensionality of the has greatly increased which worked in favor of the neural network we used here as by nature more features help the neural network model to understand the data more explicitly. But this same reason caused the other two models to have poor accuracy since these classical machine learning models are not suitable to deal with such high dimensionality of the data. Additionally, Naive Bayes model requires the assumption that predictors/records should be independent to have a good performance, it also can be explained why Naive Bayes model has lowest accuracy among all models and two datasets.

6 Conclusion

In this work, we compare a shallow Neural Network with Support Vector Machine and Naive Bayesian model on KDDcup99 and UNSW-NB15 dataset. As we expected, a low-layer neural network outperforms the other two approaches due to the big size of datasets, which partially confirms that neural network models might be a better choice for the development of intertect network intrusion detection, since the heavy traffic loads we have now. In contrast, the training of neural network models takes longer time and more computational power as the limitation of application of neural network models. In the future, we expect to improve the running time as well as the detection rate by using the models on a more diverse dataset, changing the architecture of the neural network and fine tuning the parameters, so that we can make a more comprehensive summary for the advantages and disadvantages of using neural network model in intrusion detection system researches.

Contribution

In this project we have divided the tasks among the three members. Doing so helped increase our efficiency and allowed the effective accomplishment of the project within the given time. Playing with the data helped us to identify the preprocessing required in order to make the dataset ready for application of machine learning and deep learning models.

The preprocessing of KDD Cup 99 dataset was done by Yang as following steps as the data investigation, the missing and faulty value detection, the numerical value transformation, the normalization and the one hot encoding. Furthermore, Yang is in charge of the development of modifying deep neural network model which is based on reference paper [12]. He contributes on the writing of the majority of introduction, conclusion and all the neural network associating parts in the report.

Prithila did the data visualization for UNSW- NB15. Afterwards she did the encoding on that particular dataset. One hot encoding was applied to convert the categorical features to numerical one as discussed earlier in the report. Before applying the encoding, she dealt with the imbalanced data to get rid of the problem that could occur due to different dimensionality. Other than that, she ran Naive Bayes on UNSW - NB15 dataset as well as Convolution Neural Network on that very dataset.

Zunayed ran both Naive Bayes and SVM on KDD Cup 99 dataset as well as SVM on UNSW - NB15 dataset. After running all the algorithms and getting results, it is very necessary to compare the performance of our proposed models in order to correctly evaluate them. Zunayed ran the confusion matrix for each applied model and worked on the result visualization by plotting the confusion matrix as well as classification accuracy graphs.

References

[1] J.S. Bridle, “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition,” *Neurocomputing—Algorithms, Architectures and Applications,* F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)

[2] Martellini, Maurizio; Malizia, Andrea (2017-10-30). Cyber and Chemical, Biological, Radiological, Nuclear, Explosives Challenges: Threats and Counter Efforts. Springer. ISBN 9783319621081

[3] Axelsson, S (2000). "Intrusion Detection Systems: A Survey and Taxonomy" (retrieved 21 May 2018)

[4] Tavallaee, Mahbod & Bagheri, Ebrahim & Lu, Wei & Ghorbani, Ali. (2009). A detailed analysis of the KDD CUP 99 data set. IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA. 2. 10.1109/CISDA.2009.5356528.

[5] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, 2015, pp. 1-6.

[6] Garzia, Fabio; Lombardi, Mara; Ramalingam, Soodamani (2017). An integrated internet of everything — Genetic algorithms controller — Artificial neural networks framework for security/safety systems management and support. 2017 International Carnahan Conference on Security Technology (ICCST). IEEE. doi:10.1109/ccst.2017.8167863. ISBN 9781538615850.

[7] Denning, Dorothy E., "An Intrusion Detection Model," Proceedings of the Seventh IEEE Symposium on Security and Privacy, May 1986, pages 119–131

[8] S. J. Horng, M. Y. Su, Y. H. Chen, T. W. Kao, R. J. Chen, J. L. Lai and C. D. Perkasa, “A novel intrusion detection system based on hierarchical clustering and support vector machines,” in Expert systems with Applications, vol. 38, no. 1, pp. 306–313, 2011.

[9] B. Subba, S. Biswas and S. Karmakar, “Intrusion detection systems using linear discriminant analysis and logistic regression,” 2015 Annual IEEE India Conference (INDICON), New Delhi, 2015, pp. 1-6.

[10] D. Jing and H. Chen, "SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset," 2019 IEEE 13th International Conference on ASIC (ASICON), Chongqing, China, 2019, pp. 1-4.

[11] Yin, C.L., Zhu, Y.F., Fei, J.L., et al.: ‘A deep learning approach for intrusion detection using recurrent neural networks’, IEEE. Access., 2017, 5, pp. 21954–21961

[12] Jia, Y., Wang, M., Wang, Y.G.:’ Network intrusion detection algorithm based on deep neural network’, ISSN., 2019, pp. 1751-8709

[13] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, “Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation,” discex, vol. 02, p. 1012, 2000.

[14] KDD-CUP-99 Task Description. (2020). Retrieved 15 March 2020, from http://kdd.ics.uci.edu/databases/kddcup99/task.html?fbclid=IwAR1JcWfdh-J1pLiYmhaWJIg-DXmsXZpZlp8tcFsPcTiYwshw2PDoSesRQcI

[15] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole, "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu, 2018, pp. 1-8.

[16] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, 2009, pp. 1-6.

[17] Gandhi, R. (2020). Support Vector Machine — Introduction to

Machine Learning Algorithms. Medium. Retrieved 15 March 2020, from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[18] Wafa' S.Al-Sharafat, and Reyadh Naoum "Development of Genetic-based Machine Learning for Network Intrusion Detection" World Academy of Science, Engineering and Technology 55, 2009

[19] A. Aziz, A. S., Hanafi, S. E.-O., & Hassanien, A. E. (2017). Comparison of classification techniques applied for network intrusion detection and classification. Journal of Applied Logic, 24, 109–118. doi:10.1016/j.jal.2016.11.018