INTRUSION DETECTION WITH AN AUTOENCODER AND ANOVA FEATURE SELECTOR

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Rashmi Satyal

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

INTRUSION DETECTION WITH AN AUTOENCODER AND ANOVA FEATURE SELECTOR

By

Rashmi Satyal

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall E. Nygard
Chair

Dr. Jen Li

Dr. Limin Zhang

Approved:

April 19, 2021
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

Intrusion detection systems are systems that aim at identifying malicious activities or violation of policies in a network. The problem of high dimensionality in intrusion detection systems is a barrier in processing data and analyzing network traffic.

This work aims at tackling problems associated with high data dimensionality using a feature selection technique based on one way ANOVA F-test before the classification process. It also involves study of autoencoder as a classification technique for network data as opposed to the traditional use of autoencoders in image data. Experiments have been conducted using the popular NSL-KDD dataset and the results of those experiments are compared with existing literature.

Keywords: Machine learning, feature selection, intrusion detection, autoencoder, ANOVA F-test, NSL-KDD

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to Dr. Kendall E. Nygard, my advisor, who gave me the flexibility and opportunity to find my path in this journey. Thank you for encouraging me to always work on what excited me the most and for helping me overcome obstacles that came in the way. I am deeply indebted to Dr. Jen Li and Dr. Limin Zhang for being a part of my supervisory committee, devoting time from their busy schedule.

I very much appreciate valuable inputs provided by my fellow researchers, Mostofa Ahsan and Aakanksha Rastogi while we were working on different publications. Their inputs and feedbacks have been very insightful.

Getting through my master's thesis required more than academic support and I have my parents to thank for their constant support and motivation in this journey. I am also grateful to my extended family, friends and educators who have been a source of support over the years, directly or indirectly. My family in the US have been very supportive in the years of my Master's and my thesis would be incomplete without thanking them.

DEDICATION

I dedicate this thesis to the most important people in my life, my parents – Prof. Dr. Vikash Raj Satyal and Mrs. Manju Risal Satyal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A data breach of the airline's technology vendor made it to the headline of many online news portal recently. The multinational IT company 'Société Internationale de Télécommunications Aéronautiques' (SITA) gets passenger information such as membership number, tier status and even credit card information, to name a few, from international airlines including Singapore Airlines, Finnair, Malaysian Airlines etc. Although it was states by SITA officials that the data breach did not impact sensitive passenger information, it is not hard to imagine the dire consequences SITA and the associated airlines would have had in that case [1].

In today's world of connected devices, security of the network is of critical importance. Networking revolution in recent times has provided limitless connectivity increasing, greatly, opportunities for malicious attacks. Unauthorized access and malicious activities are a great threat to confidentially, integrity, and availability that form the information security triad. Preventing such unwanted access to network devices is becoming more challenging and important. While systems cannot be completely shielded from attacks by intruders, such attempts can be detected and analyzed. Intrusion detection systems (IDSs) are reactive agents that detect such vulnerabilities in the network.-The role of an Intrusion Detection System (IDS) is to detect abnormalities caused by an unauthorized reach into the network and send alerts. Depending on the goal of the system, IDSs can be differentiated into different types. Two main types of intrusion detection systems that are often talked about are signature-based intrusion detection systems and anomaly-based intrusion detection systems.

Trying to overcome the problems faced with the usage of signature-based intrusion detection systems, researchers shifted their focus to anomaly detection approaches. The baseline dataset that was being used for the analysis of different anomaly-based intrusion detection systems

was the KDD'99 dataset. A group of four researchers – Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani conducted some statistical analysis on the KDD'99 dataset and found some note-worthy issues with it that would result in inept performance of the IDS using it [2]. Two of the most concerning issues with this publicly available KDD'99 dataset listed in their work were:

- The presence of a huge number of redundant records in the dataset (78% and 75% redundancy in the training and test data respectively).
- Unproportionate distribution and skewness in KDD'99 test dataset resulting in suspiciously high accuracy rate even with the use of simple learners. This makes it a poor indicator of the ability of the classifier to be used as a tool in anomaly-based intrusion detection systems.

To overcome these problems, they proposed a solution in the form of a new dataset: the NSL-KDD dataset. At present, the NSL-KDD dataset has gained high popularity and is considered one of the benchmark datasets for anomaly detection system evaluation. The advantages it has over its predecessor are as follows [3]:

- There is no redundancy in the training set records. This helps remove bias of the classifier towards duplicate records as duplicate data in training set implies that the 'weight' of that record is doubled making is twice as important in the model's perception.
- It also does not include redundant records in the test set. This eliminates the problem that occurs when a learning algorithm has better detection rate of frequent data, giving an inflated sense of the model's overall performance.
- It has a reasonable number of records in both training and test sets.

When a high-dimension dataset such as the NSL-KDD dataset is being used, using some feature engineering techniques enhances the performance of the system. Feature engineering is the process of modifying features to better encapsulate the nature of a machine learning problem. Any input to a machine learning model, whether it is supervised or unsupervised, includes instances that consist of features. Features are what provide information about instances. Depending on the problem at hand, identifying features that are more relevant to problem can make analyzing the dataset more efficient. The idea of feature engineering is to add to the enhancements made in performance by tuning the machine learning algorithm itself [4]. Even after tuning the hyperparameters in a learning algorithm, performance can be further enhanced by using feature engineering methods. Feature selection – a form of feature engineering helps determine those features in a high-dimensional dataset that are most relevant to the task at hand and makes increases the time-efficiency of the learning algorithm.

The aim of this research work is two-fold. The first objective is to develop an intrusion detection system using a special machine learning technique known as an autoencoder to detect network anomaly. Performance evaluation of the proposed model is done using the NSL-KDD dataset introduced above. The second objective of the research is to analyze the significance of ANOVA F-test as a feature selection method for network intrusion data. The future prospect of this work is to help identify techniques that can be used in real-time scenarios efficiently.

As mentioned above, the machine learning algorithms of interest for this research is an autoencoder. Autoencoders are a learning technique, specifically artificial neural networks (ANN), that are capable of learning input data representation using compression followed by reconstruction. Building blocks of an autoencoder can be seen in figure 1 below.

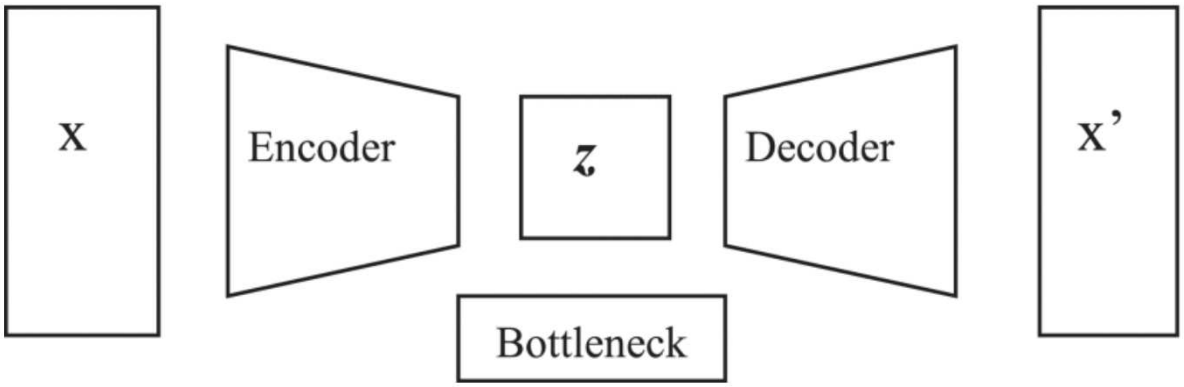

Figure 1. Autoencoder [30].

Autoencoders basically work by encoding input data at the 'encoder' block, compressing data at the 'bottleneck' (middle) layer and reconstructing the encoded input at the 'decoder' block. While autoencoders are vastly used in image compression tasks, this research work demonstrates the use of autoencoders for intrusion detection.

Now-a-days, many open source tools and frameworks are available that make the task of running experiments easy for programmers. One such tool is Keras. Keras, built on top of Tensorflow, provides APIs (Application Programming Interface) for building neural networks. It simplifies the use of Tensorflow and helps to translate ideas into implementation with least possible delays. The major advantage of using Keras is it helps developers avoid low-level computations. It has built-in functions that makes the training and implementation of autoencoders easy.

The rest of the thesis is organized as follows. Section II sheds some light in the review of previous literature on which the research work is based. Section III goes into detail explanation of the methodology used in this study. It dives into more detailed analysis of the dataset used. It then gives a description of the tools and technologies used to conduct experimental runs. Subsection III.C. explains the preprocessing methods used on the data to make it usable by the machine

learning model, followed by subsections III.D. and III.E. that discuss the feature selection approach and the machine learning model used, respectively. The outcome of the experiments has been explained in section IV. Finally, in Section V. the conclusions that can be drawn from this research work have been discussed.

II. LITERATURE REVIEW

Machine learning methods traditionally used in anomaly-based intrusion detection system include Naïve Bayes (NB) classifiers, Support Vector Machine (SVM), k-Nearest Neighbors (KNN) and decision tree classifiers. In the evaluation done by M. C. Belavagi and B. Muniyal [5] they have used the NSL-KDD dataset with four different classifiers: logistic regression, support vector machine, guassian naïve bayes and random forest classifier and presented a comparison of accuracies obtained in each case. Logistic regression, a statistical classifier that uses a logistic function to predict probability of occurrence of any event is demonstrated to be 84% accurate in the NSL-KDD dataset. SVM, where data points are plotted in an n-dimensional space (n being the number of features) and Gaussian Naïve Bayes classifier, a classifier based on Bayes theorem are found to have 75% and 79% accuracy, respectively. Random forest, a classifier that uses votes from multiple decision trees to make a prediction, outperforms the other three classifiers in this study with a very high accuracy of 99%. With the intent of evaluating some non-classical machine learning approaches, Z. Li, A. L. G. Rios, G. Xu, and L. Trajkovic deployed two deep learning Recurrent Neural Networks (RNNs) – Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) along with Broad Learning System (BLS) on the NSL-KDD dataset [6]. They concluded that LSTM and GRU deep neural networks and the BLS model all achieved comparable performance in terms of accuracy which were close to 83% in all cases. The experiment performed by B. Ingre and A. Yadav [7] on the NSL-KDD data shows results of conducting binary class and five-class classification using Artificial Neural Network (ANN). They have performed feature elimination based on observation of the dataset, removing four features as almost values in those features were zero. They have also eliminated eight features based on previous research [8]. Using tansig transfer function, Levenberg-Marquardt (LM) and Broyden–Fletcher–Goldfarb–Shanno

(BFGS) quasi-Newton Backpropagation, with 21 neurons in the hidden layer and changing output layer's neurons to 2 for two class and 5 for five class classification, they reported the model accuracy to be 81.2% for intrusion detection (binary classification) and 79.9% for attack type classification (five class classification).

Autoencoders are a type of artificial neural network that are capable of learning the representation of input data and help reduce dimensionality from the original input. This makes it autoencoders the model of choice for image compression. Much time and effort has been spent on the evaluation of autoencoder as a compression tool. The use of Convolutional AutoEncoders (CAEs) was projected to have better performance than traditional transforms for lossy compression of image data in the work by Z. Cheng, H. Sun, M. Takeuchi, and J. Katto [9]. The CAE network was supplied a subset of ImageNet dataset and the autoencoder was configured to use the Parametric Rectified Linear Unit (PreLu) activation function with Adam optimizer. The use of autoencoders by C. C. Tan and E. Eswaran in medical image compression (specifically mammograms) [10] shows promising results in terms of Mean Squared Error (MSE) and Structural Similarity (SSIM) index for different combinations of hidden layers. Although autoencoder has mainly seen popularity in image compression applications, researchers are starting to inquire if it can be used as a viable model for intrusion detection. M. Gharib, B. Mohammadi, S. H. Dastgerdi and M. Sabokrou have presented AutoIDS as an autoencoder based IDS solution[11]. By stacking a sparse and a reconstruction-error based autoencoders, they have created a model that was able to achieve an accuracy of 90%.

Feature selection is a popular data preprocessing step in machine learning applications. Removing unwanted and irrelevant features from a dataset has been found to significantly improve the efficiency of a leaning model. N. O. F. Elssied, O. Ibrahim and A. H. Osman demonstrated the

use of one-way ANOVA F-Test[12] as an efficient method of feature selection in email-spam classification. They combined this feature selection method with SVM and concluded that this approach provided better performance in terms of accuracy, computation time and false positive rates. A study of different feature selection methods by G. Chandrashekar and F. Sahin[13] presents different filter and wrapper methods such as correlation criteria, genetic algorithm, and sequential floating forward selection (SFSS). They evaluate the use of these feature selection methods on different datasets and conclude that while feature selection can improve the performance of a model the, every algorithm will behave differently for different data and applications. Using dataset with mixed datatypes (categorical and non-categorical), L. A. A. Almeida and J. C. M. Santos[14] found that the use of VarianceThreshold function provided by the Python library Scikit-learn gave good results in conjunction with SVM.

III. METHODOLOGY

In this section, the description of the approaches to identify anomaly using the NSL-KDD dataset has been discussed. As mentioned in section I, the work is divided into two parts. In the first part, the machine learning model learns to differentiate between ‘normal’ and ‘attack’ data without the implementation of any feature selection methods. A high-level flow chart of this process is shown in figure 2 below:

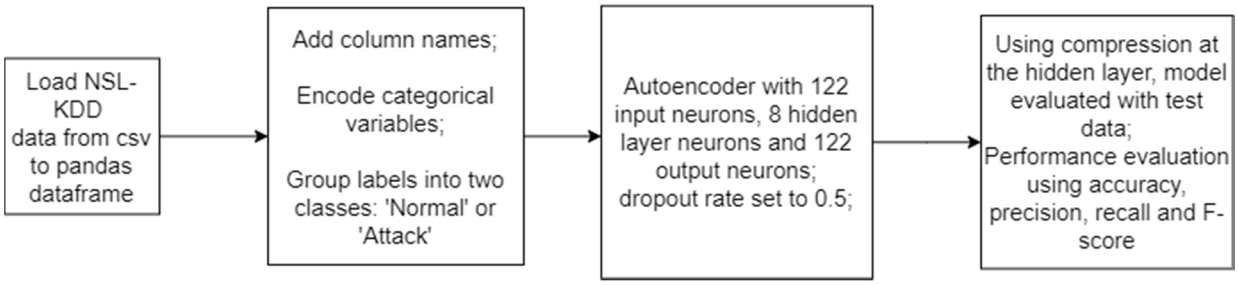

Figure 2. Flow chart – normal vs. attack classification

In the second part, the attack labels are separated into Normal, DoS, Probe, R2L and U2R. Then, the dataset is divided such that at one time it consists of Normal and one other attack type. Feature selection is applied to this subset and it is then fed to the machine learning model. This process is repeated for all other attack types. A high-level flow of this process is shown in the figure 3 below:

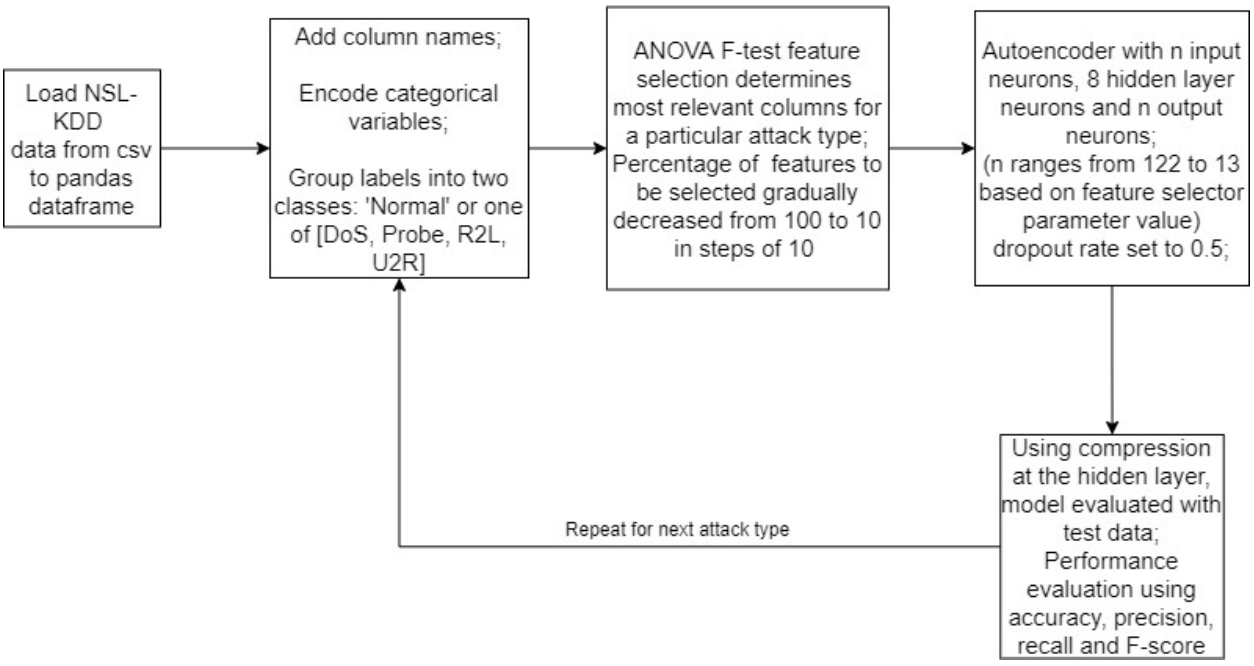

Figure 3. Flow chart – feature selection

The rest of this section dives deeper into description of data, tools and technologies used, data preprocessing steps taken, description of the machine learning model and the feature selection method applied.

3.1. Data description

The NSL-KDD is an update and improvement to the KDD'99 dataset that that was developed for the KDD Cup competition in 1999 [15]. These datasets are publicly available and are very widely used for IDS experiments. The data is primarily internet traffic consisting of 43 features per record, of which the last two are class (specific attack type or normal) and score (severity of traffic input) [16]. The class column provides information on whether the record is considered normal or is a member of one of four attack classes – Denial of Service (DoS), Probe, Remote-to-Local (R2L) or User-to-Root (U2R). There are 14 attack types. A mixture of categorical (nominal), binary and numeric variables are in the feature set. Each record has basic, content-

related, time-related, and host-based features [17]. The attributes of this dataset are listed in table 1 below.

Table 1. NSL-KDD dataset attributes[18]

Attribute Type	Attribute Names
Basic	Duration, Protocol_type, Service, Flag, Src_bytes, Dst_bytes, Land, Wrong_fragment, Urgent
Content related	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_hot_login, Is_guest_login
Time related	Count, Srv_count, Serror_rate, Srv_serror_rate, Rerror_rate, Srv_rerror_rate, Same_srv_rate, Diff_srv_rate, Srv_diff_host_rate
Host based traffic	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

A description of the features in the training and test datasets are presented in appendix table A1.

The NSL-KDD dataset and the KDD dataset are derived from the dataset provided by 1998 DARPA (Defense Advanced Research Projects Agency) Intrusion Detection Evaluation Program [28]. The 1998 DARPA dataset is based on network audit logs that have attack names based on attack signatures. For example, in 'land' attack the attack signature is the presence of identical source and destination addressed in network packets [29]. This method of assigning class label is also true in the NSL-KDD dataset as it is ultimately derived from the 1998 DARPA dataset.

3.2. Tools and technologies

Python: Python programming language is being used more and more for scientific computing. A survey conducted by *Stack Overflow* in 2020 shows that Python is the 4th most

popular programming language among programmers [19]. This consistent popularity over the years can partly be credited to the extensive and powerful packages python provides for data analysis. Some of the important packages used in this research have been described in the following sections.

Anaconda Navigator: Anaconda is a flexible tool that among other features, gives access to different integrated development environments (IDEs). Spyder – the Python IDE in Anaconda was chosen for the project due to it easy-to-use editing, testing and debugging features[20]. The use of Anaconda Navigator greatly simplified the task of installing and managing packages.

Scikit-learn: Scikit-learn is an important framework for data science and machine learning in Python. Written mostly in Python, the scikit-learn project is an active project with new enhancements being added constantly [21].

Pandas: One of the reasons Python has become the programming language of choice for statistical computing is Pandas. The Pandas library is a data structure tool that provides various functionalities for data manipulation and analysis[22].

NumPy: NumPy, another Python library that deals with data structures contrasts from Pandas in that it works with numerical data where Pandas works with tabular data. As this research project is data intensive in nature, use of the NumPy library plays a significant role in handling the dataset.

Matplotlib: This library helps users create simple 2D plots in Python with just a few commands. All plots in this project have been created using Matplotlib.

Keras: A general introduction to Tensorflow and Keras have been provided in section I above. In this project, Keras has been used with Tensorflow as the backend to implement autoencoder.

3.3. Data preprocessing

Column names: The first step in the data preprocessing step is naming the columns in the training and test datasets.

Generalizing labels: The test data set consists of an additional 15 attack types that are not present in the training dataset. Thus, it is useful to provide more generalized labels to the attacks to train the model. The 37 attack types mentioned before have been generalized into 4 categories:

- Denial of Service (DoS) attacks: This attack group is caused when attackers make the network resources too busy so that legitimate users seeking access to the resources cannot do so.
- Remote-to-Local (R2L) attack: R2L is a malicious attack that occurs when someone who doesn't have an account in a local machine sends packets to the machine over a network.
- User-to-Root (U2R) attacks: A root user has elevated privileges in a computer. In a U2R attack, a normal user in a computer exploits vulnerabilities to gain root access to the system. This can be done by password sniffing, dictionary attacks or even through social engineering.
- Probe attacks: A probe attack is unique in the sense that the attacker wants the attack to be detected by the target system. Once the target system detects the attack and reports it to public repositories, attacker is able to confirm that the system is indeed being monitored (in this case, the attacker is also part of the public repository). A detailed analysis of the alert reported by the target system reveals additional system information such as make and model of the target's IDS, target's network topology

and how the exploit was detected. This information is in turn exploited by the attacker. [27]

Table 2 shows the grouping of these attack types into the 4 labels mentioned above:

Table 2. Generalized attack labels

Generalized labels	Attack labels in original dataset
Denial of Service	snmpgetattack, back, land, neptune, smurf, teardrop, pod, apache2, udpstorm, processtable, mailbomb
Remote-to-Local	snmpguess, worm, httptunnel, named, xlock, xsnoop, sendmail, ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
User-to-Root	sqlattack, buffer_overflow, loadmodule, perl, rootkit, xterm, ps
Probe	ipsweep, nmap, portsweep, satan, saint, mscan

Out of the four classes mentioned above, DoS and Probe attacks have a higher count in the training and test data whereas R2L and U2R are underrepresented. Table 3 below highlight the distribution of these attack types in the training and test datasets:

Table 3. Attack distribution in NSL-KDD dataset

Attack labels	Total instances in training data	Percentage in training data	Total instances in test data	Percentage in test data
DoS	45927	77.48%	7460	56.78%
Probe	11656	19.66%	2421	18.43%
R2L	1642	2.77%	3191	24.28%
U2R	52	0.09%	67	0.51%
Total	59277	100%	13139	100%

There is a total of 125973 rows in the training data and 22544 rows in the test dataset. From the table above, we can see that analysis on the DoS and Probe attack types are more significant than that on R2L and U2R attacks.

Feature elimination: The 'Score' field from the dataset has been eliminated. The field has been eliminated at the very beginning when the training and test csv files are loaded, because it does not add to the analysis.

Target field: The target field 'label' has been dropped in the preprocesing step and replaced with a 'class' field. In the new 'class field, attack labels have been replaced with the generalized labels mentioned in table 2 above.

One-hot encoder: An autoencoder cannot perform on non-numeric data. As the NSL-KDD dataset used consists of categorical data in the *protocol_type*, *service* and *flag* fields, one-hot encoder is used in the data-preprocessing step to convert them into appropriate numerical representation. This method converts an attribute with N possible categories into N distinct features. For example, in the NSL-KDD dataset, the *protocol type* attribute has 3 possible values – Internet Message Control Protocol (ICMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). One-hot encoding converts this attribute into three feature columns as shown in figure 4.

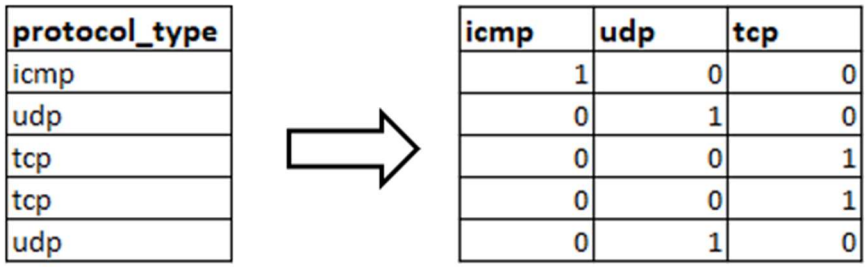

Figure 4. One-hot encoding used in protocol type attribute.

For the remining features, StandardScalar function has been used, which is described in the following section.

StandardScalar: Values in different features within a dataset can have different scales making it hard to compare, analyze and visualize data. Scaling is a standard data preprocessing

step because features that are fed to a machine learning model as inputs at different scales do not contribute equally to the model fitting and learning processes which gives rise to an unwanted bias. In this work, StandardScalar function from the sklearn.preproessing package has been used. This scalar function removes the mean from the feature and scales its values to unit variance.

3.4. Classification

For the classification task, dropout autoencoder has been used for model training. In an autoencoder, dropout is an efficient method to prevent overfitting. After encoding the features with one-hot encoder, the dataset now contains 122 features. Input layer in the sparse autoencoder thus has 122 neurons. The input layer is followed by a dropout layer configured at a dropout rate of 0.5. This dropout layer randomly sets input units to 0 based on the dropout rate defined [23] which is what prevents overfitting. To make sure the overall inputs are unchanged, the input units that are not set to 0 are then scaled to 1/(1-*rate*) , which is 1/(1-0.5) in this case. The dropout layer is followed by a hidden layer with 8 neuron. 8 neurons in the hidden layer result in an autoencoder with a compression rate of 122/8. The final layer is the output layer that again has 122 neurons.

ReLu (Rectified Linear Unit) function along with Adagrad optimizer has been used in the model. Choice of ReLu activation function overcomes vanishing gradient problem and it allows the machine learning model to speed up learning. ReLu is the most commonly used activation function in artificial neural networks because of it's simplicity. For each experiment run, the model was trained for 10 epochs with a batch size of 100 and a validation split of 0.1.

3.5. Feature selection

ANOVA (**An**alysis **o**f **Va**riance) is a statistical technique that incorporates hypothesis testingthat, in the simplest sense, analyzes the difference among means among different groups. One-way ANOVA is a method determines relationship between predictor and a response variables.

The one-way ANOVA F-test statistics has been used in this work to eliminate unimportant features from the dataset. This algorithm takes one attribute into consideration at a time to see how well each predictor (feature) by itself predicts the target variable (output) [12]. The f_value between the predictor and target is calculated which is used as a parameter in the SelectPercentile method provided by scikit-learn. SelectPercentile then returns the percent of features to keep which is defined in its second parameter 'percentile'. This feature selection method is used to transform the training data. For every attack type (DoS, Probe, R2L and U2R), the most meaningful 10% of the total features (i.e. 13 features) have been extracted in separate experimental runs. Only these 13 features are fed as input to the autoencoder for learning.

IV. EXPERIMENTAL RESULTS

The performance of each experimental run is evaluated in terms of accuracy of the machine learning model. Accuracy is a metric that measure how correctly the model classifies a data point and is calculated as follows:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

As the model used in this project follows binary classification, the accuracy can also be viewed in terms of true and false positives and negatives as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positive, TN = True Negative, FP = False Positive and FN = False Negative.

The positive and negative values for each run can be visualized easily using a confusion matrix. Accuracy measure alone, however, does not provide complete information of how good the model performance is. A deeper analysis of these positive and negative values gives a broader perspective on the model performance. Performance measures that provide additional insights are: precision, recall and f-score. The ratio of true positive values to all positive values gives *precision*.

$$Precision = \frac{TP}{TP + FP}$$

Recall, another metric for performance evaluation, is the fraction of retrieved instances among all relevant instances[24]. Mathematically, recall can be defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

F-score is the harmonic mean for precision and recall. A good f-score implies a low false positive and low false negative rate.

The final output of the model is a performance table that shows how the model behaves in terms of accuracy, precision, recall and f score when presented with features of different nature and dimensions. These metrices are based on values in the confusion matrix for each case.

4.1. Result of experiment conducted on the basis of normal vs. attack label

Figure 5 below show the number of correctly classified and misclassified normal and attack labels in the form of a confusion matrix.

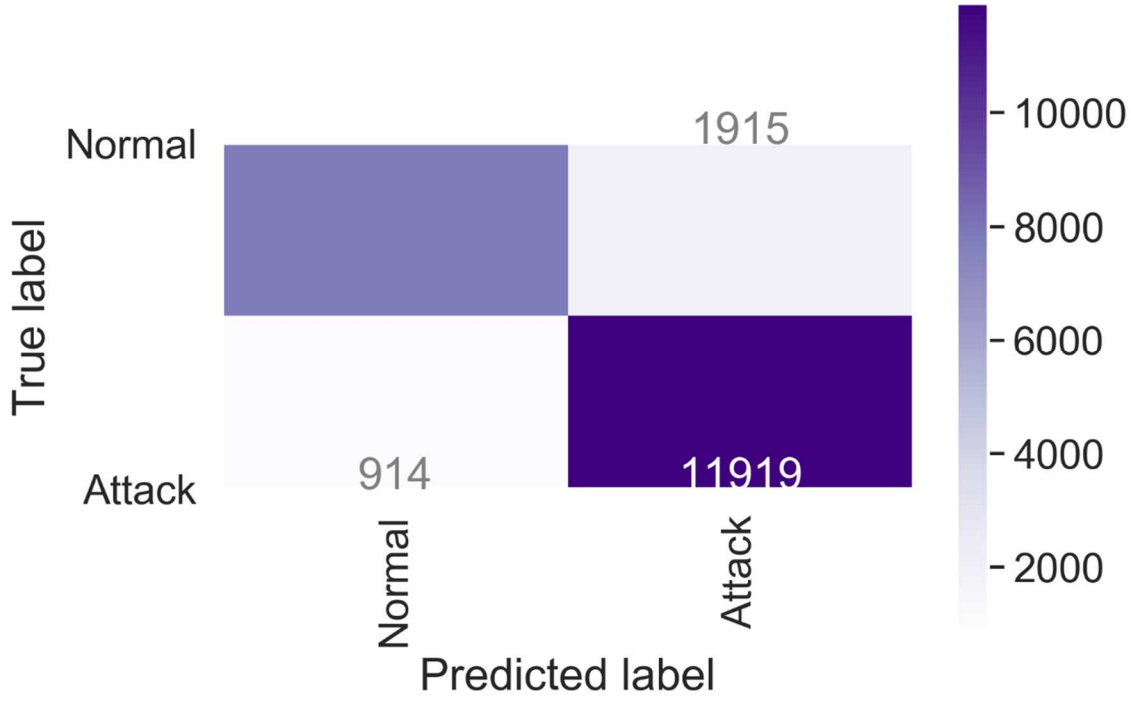

Figure 5. Confusion matrix – normal vs. attack label

Table 4 below shows performance of the machine learning model when it is tasked with distinguishing between a normal and an attack data:

Table 4. Performance evaluation of normal vs. attack classification

Accuracy (%)	Precision	Recall	F1_score
88.76	0.852	0.971	0.908

The table above suggests that the model is a good binary classifier for threat detection as all four performance metrices are high.

4.2. Result of experiment conducted on the basis of normal vs. DoS label

Figure 6 below illustrates the number of correctly classified and misclassified normal and DoS labels in the form of a confusion matrix.

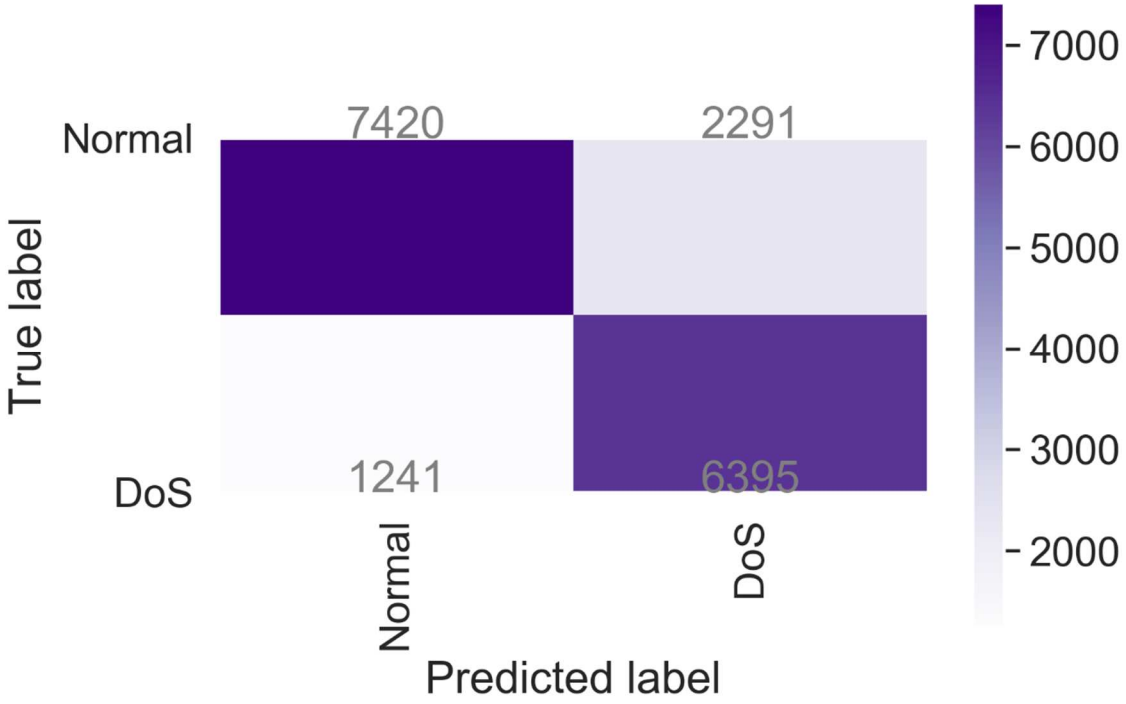

Figure 6. Confusion matrix – normal vs. DoS label

Table 5 below depicts performance of the machine learning model when it is tasked with distinguishing between a normal and a DoS attack data:

Table 5. Performance evaluation of normal vs. DoS classification

Accuracy (%)	Precision	Recall	F1_score
84.0	0.800	0.760	0.780

The performance table for Normal vs. DoS classification indicates that the model can identify a DoS with 84% accuracy. The false positive and false negative rates are also low as shown by the high precision, recall (and hence f score) values.

The list of features selected by the feature selection method for this dataset are: ['logged_in', 'count', 'serror_rate', 'srv_serror_rate', 'same_srv_rate', 'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'service_57', 'flag_2', 'flag_9']

4.3. Result of experiment conducted on the basis of normal vs. probe label

The number of correctly classified and misclassified normal and probe attack labels is displayed in figure 7 below, in the form of a confusion matrix.

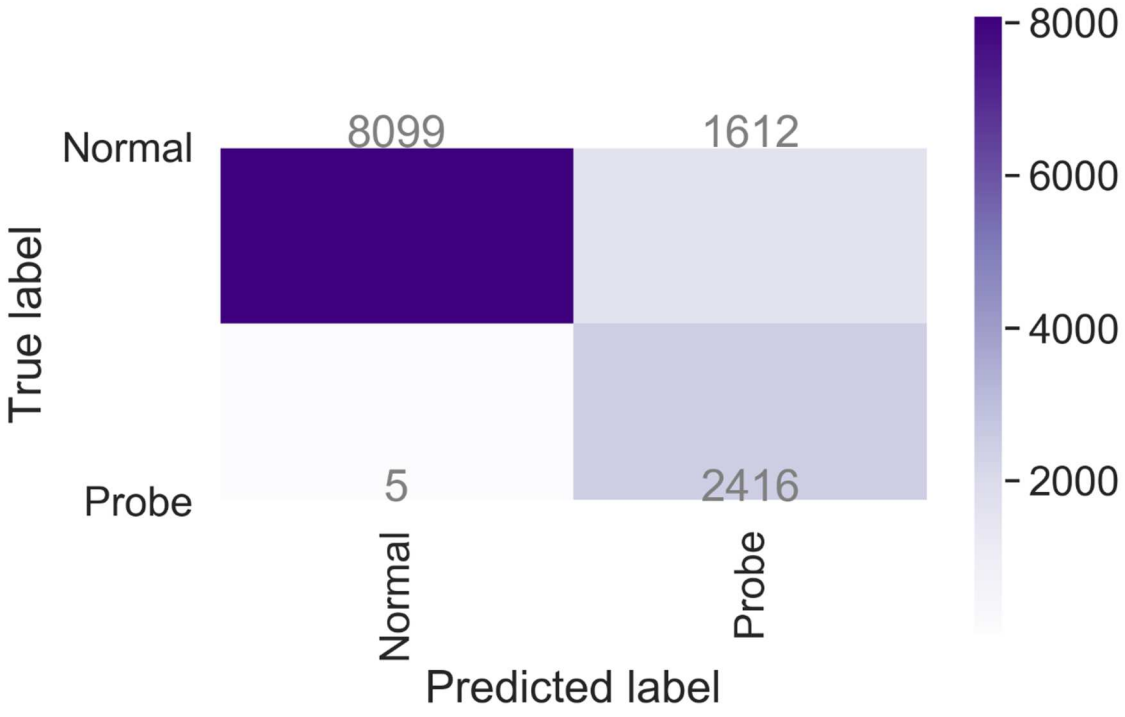

Figure 7. Confusion matrix – normal vs. probe label

Table 6 below shows performance of the machine learning model when it is tasked with distinguishing between a normal and a probe attack data:

Table 6. Performance evaluation of normal vs. probe classification

Accuracy (%)	Precision	Recall	F1_score
87.67	0.618	0.997	0.763

Overall performance of the classifier shows good result for Probe attack detection.

The list of features selected for this dataset are:

['logged_in', 'rerror_rate', 'srv_rerror_rate', 'dst_host_srv_count', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'dst_host_rerror_rate', dst_host_srv_rerror_rate', 'protocol_type_2', 'service_68', 'service_85', 'flag_2']

A closer look at the features selected by the feature selection algorithm in both DoS and Probe attacks suggests that *logged_in* attribute, which indicates a successful or an unsuccessful login attempt, is a good indicator for both attack types. From above, it can been observed that DoS attacks are mainly characterized by *serror_rate* whereas Probe attacks are mainly characterized by *rerror_rate*. These attributes depend on the *flag* attribute where serror_rate is related to *flag* status S0, S1, S2 and S3 whereas *rerror_rate* is related to *flag* status REJ. The way a connection ended seems to be another good indicator of an attack. A *flag* status 'SF' indicates a normal SYN/FIN completion, which is found to not be a characteristic of an attack from the above analysis. S0, S1, S2, S3 and REJ status occur in the following circumstances [26]:

- **S0 (state 0)**: initial SYN seen but no reply.
- **S1 (state 1)**: connection established (SYN's exchanged), nothing seen further.
- **S2 (state 2)**: connection established; initiator has closed their side.
- **S3 (state 3)**: connection established; responder has closed their side.
- **REJ (connection rejected)**: initial SYN elicited a RST in reply;

dst_host_srv_count is also common in both DoS and Probe attacks. It is the number of connections that have the same port number.

4.4. Result of experiment conducted on the basis of normal vs. R2L label

The number of correctly classified and misclassified normal and R2L attack labels is presented in figure 8 below, in the form of a confusion matrix.

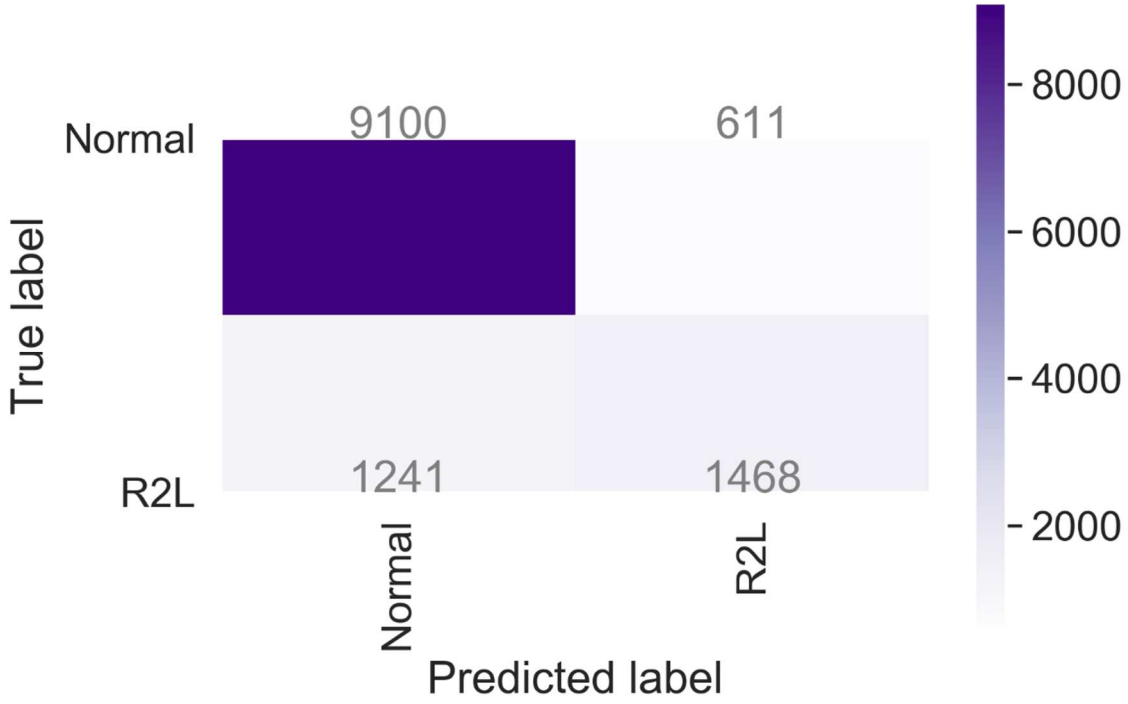

Figure 8. Confusion matrix – normal vs. R2L label

Table 7 below shows performance of the machine learning model when it is tasked with distinguishing between a normal and an R2L attack data:

Table 7. Performance evaluation of normal vs. R2L classification

Accuracy (%)	**Precision**	**Recall**	**F1_score**
84.77	0.6931	0.542	0.608

The list of features selected for this dataset are:

['src_bytes', 'dst_bytes', 'hot', 'num_failed_logins', 'is_guest_login', 'dst_host_srv_count', dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'service_31', 'service_54', service_57', 'service_58', 'flag_3']

4.5. Result of experiment conducted on the basis of normal vs. U2R label

Figure 9 below illustrates the number of correctly classified and misclassified normal and U2R attack labels in the form of a confusion matrix.

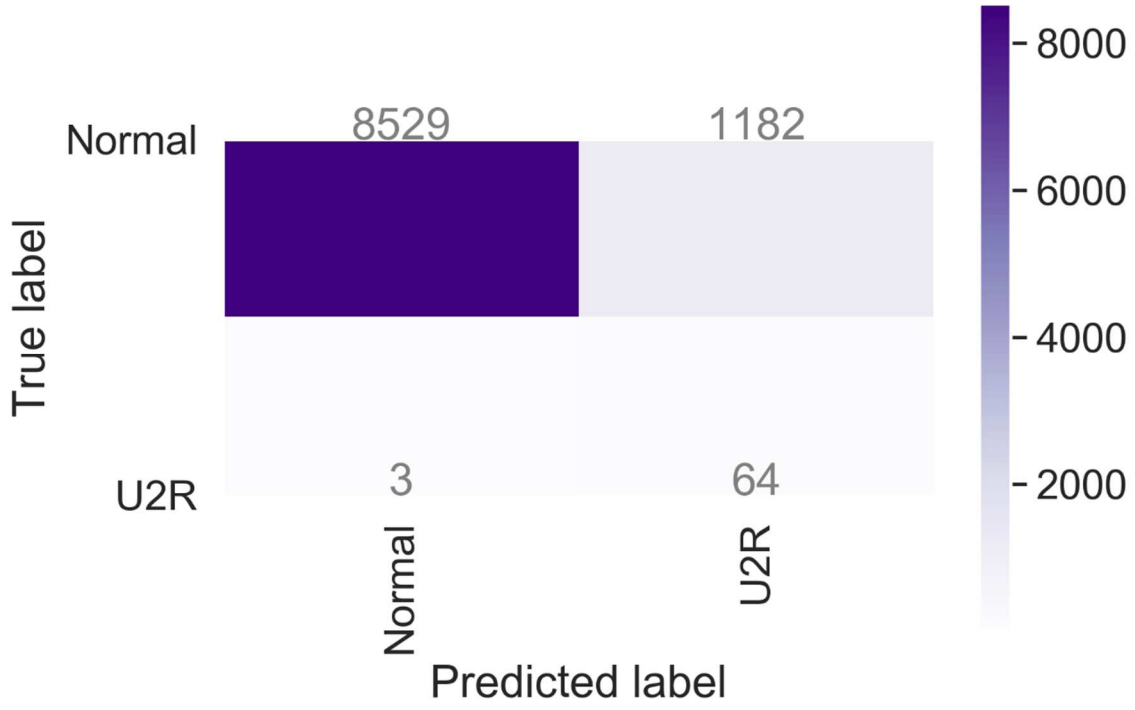

Figure 9. Confusion matrix – normal vs. U2R label

Table 8 below shows performance of the machine learning model when it is tasked with distinguishing between a normal and a U2Rattack data:

Table 8. Performance evaluation of normal vs. U2R classification

Accuracy (%)	Precision	Recall	F1_score
85.21	0.0418	0.940	0.080

The low precision and high recall values in this classification indicate unbalanced data and that most of the predicted labels are incorrect when compared to the training labels. This shows that just high accuracy is not an indicator of the overall model performance.

The list of features selected for this dataset are:

['urgent', 'hot', 'root_shell', 'num_file_creations', 'num_shells', 'srv_diff_host_rate', st_host_count', 'dst_host_srv_count', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'service_54', 'service_57', 'service_89']

The surviving features for specific attack, 13 features with the best ANOVA F-Test scores are tabulated in table 9 below:

DoS	Probe	R2L	U2R
logged_in	logged_in	src_bytes	urgent
count	rerror_rate	dst_bytes	hot
serror_rate	srv_rerror_rate	hot	root_shell
srv_serror_rate	dst_host_srv_count	num_failed_login	num_file_creations
same_srv_rate	dst_host_diff_srv_rate	is_guest_login	num_shells
dst_host_count	dst_host_same_src_port_rate	dst_host_srv_count	srv_diff_host_rate
dst_host_srv_count	dst_host_srv_diff_host_rate	dst_home_same_src_port_rate	dst_host_count
dst_host_same_srv_rate	dst_host_rerror_rate	dst_host_srv_diff_host_rate	dst_host_srv_count
dst_host_serror_rate	dst_host_srv_rerror_rate	Service_31	dst_host_same_src_port_rate
dst_host_srv_serror_rate	protocol_type_2	Service_54	dst_host_srv_diff_host_rate
service_57	service_68	Service_57	service_54
flag_2	service_85	Service_58	service_57
flag_9	flag_2	Flag_3	service_89

Figure 10. List of surviving features for specific attack types

4.6. Hyperparameter tuning

In order to determine the optimal value for the ‘percentile’ parameter, experiments were conducted multiple times reducing the number of selected features in each run. Results of these

experiments in terms of accuracy is shown in figure 10 below. Based on this graph, the value of percentile was set to 10. This resulted in the selection of the best 10% of features (i.e. 13 features) that were fed to the classifier for further processing.

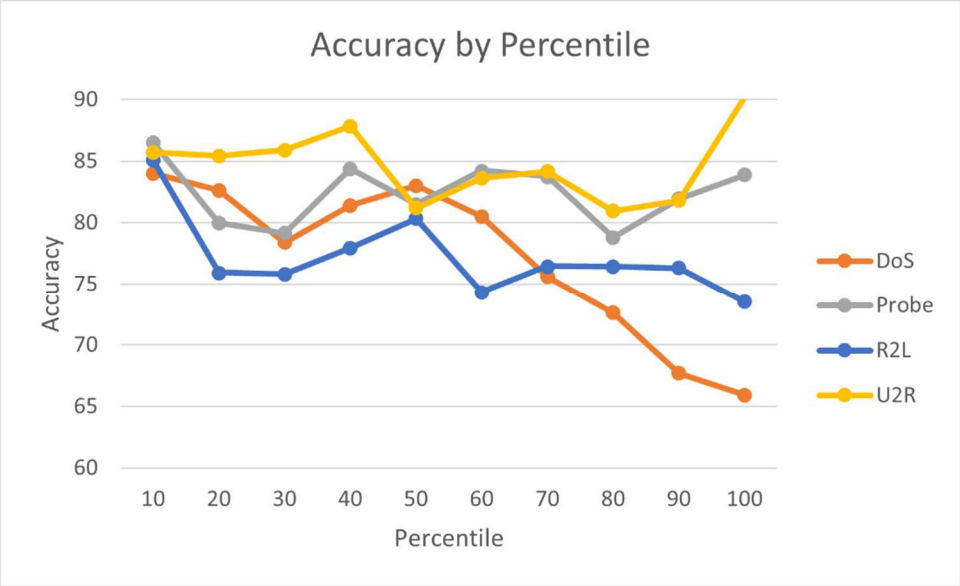

Figure 11. Accuracy vs. percentile chart for hyperparameter tuning

V. CONCLUSION

The first conclusion that can be drawn from this analysis is that the use of autoencoder for intrusion detection is effective as the two-class classification (Normal vs. Attack) accuracy is pretty high – close to 90%. This is better in comparison to the performance of traditional classifiers using the same dataset such as Gaussian Naïve Bayes and SVM but not as good as random forest algorithm. The performance is also slightly better than that of LSTM and GRU used by Z. Li, A. L. G. Rios, G. Xu, and L. Trajkovic.

Performance evaluation of Normal vs. DoS and Normal vs. Probe attacks in table 4 and 5 respectively show that the classifier can detect these attacks with an accuracy of 84% and 87.67% respectively. Majority of attacks in the NSL-KDD dataset are the DoS and Probe attacks and the machine learning model used in this research can identify these attacks with high recall. In an IDS, the cost of misclassifying an attack is very high, hence high recall value is an indicator of good model performance. When an IDS fails to identify an attack, the security personnel will have no idea an attack occurred and will not be prepared for the consequences. However, when an IDS misclassifies a normal traffic as attack, it increases the cost of dealing with the false alarm. Thus, high false negative and high false positive rates are both significant when looking at the performance of an IDS. Low values of recall in R2L and precision in U2R attacks in this research suggest the need of further enhancement of the model for these attack types.

As this method uses just a single hidden layer with 8 neurons, the training the model is not time-consuming, and it was easy to perform multiple experimental runs in a short period of time. The hyperparameters can be easily tweaked to fine-tune the model. With all features in the original dataset, the training time at epoch was noted to be 6s. Although the model trains quickly on the NSL-KDD dataset, the use of feature selection was found to reduce the training time in each epoch

by 1 second. The results show better performance in comparison to methods such as LSTM that has a training time of 4481.73s and GRU that has a training time of 1108.31s in the same dataset [6]. This can prove to be very significant when this model is used for real-time network threat monitoring and opens a possibility for the use of this application in near real-time scenario.

The use of a single hidden layer in the autoencoder was sufficient in this project. Stacking is a powerful concept in autoencoders that helps it learn more complex coding (representation of input data). However, it is important not to use more hidden layers than necessary as it can negatively impact the learning process and makes it unlikely for the model to generalize well to new data instances [25]. The use of one hidden layer in this research, which is based on previous literature, was found to be good enough and consistent with earlier studies.

Using just a small set of features, modest accuracy can be achieved. The accuracy vs. percentile chart in figure 10 shows that even though there is information loss when a subset of features is selected, the performance of classifier is decent. This finding can be used in datasets with higher volume of data to reduce training time. Analysis of DoS and Probe attacks (which make up the majority of the attacks in the NSL-KDD dataset) show that some features such as the login indicator and connection status (*flag*) are more useful in identifying attacks in network connection.

Thus, this research work demonstrates that the use of a dropout autoencoder with ANOVA feature selection algorithm is useful for anomaly detection in intrusion detection systems and the viability of this approach is strongly justified through experiments.

5.1. Limitations

Interaction between features might be different when a different group of features are selected. As there are so many features in the dataset, we can end up with a large number of

combination of features. The feature selection method used in the work is based on previous literature, but it is also possible that some other method that investigates interaction of features in a completely different way gives a more effective result than the one presented. It cannot be claimed that a particular feature selection method is the best one that can be used for a certain dataset.

An inherent problem with the NSL-KDD dataset is the class imbalance problem that can be seen in the case of R2L and U2R attacks. Table 3 shows that R2L attacks make up only 2.77% of the total attacks in training data whereas less than 0.1% of the total attacks in training dataset consists of U2R data resulting in a severe underrepresentation of these attacks. Even when compared to the number of normal instances these classes have a low count (total normal labels in training data is about 53% of the total data) which leads to unreliable results in binary classification of normal vs. R2L and normal vs. U2R attacks. Machine learning model more targeted towards improving the efficiency of an IDS in detecting these sparse attacks, as proposed by M. R. Parsaei, S. M. Rostami and R. Javidan in their study [31] can be implemented as a viable alternative to eliminate this problem. Random over-sampling and under-sampling can be used as machine learning approach to tackle the problem of class imbalance. Random over-sampling duplicates rows randomly in minority class whereas random under-sampling deletes rows randomly in majority class to attain a more balanced dataset distribution. Both of these techniques can be repeated until the desired distribution is achieved.

REFERENCES

[1] "Sprawling Cyberattack Breaches Several Airlines | Threatpost." https://threatpost.com/supply-chain-cyberattack-airlines/164549/ (accessed Mar. 19, 2021).

[2] W. Lu, A. A. Ghorbani, M. Tavallaee, and E. Bagheri, "A detailed analysis of the KDD CUP 99 data set A Detailed Analysis of the KDD CUP 99 Data Set Tavallaee A Detailed Analysis of the KDD CUP 99 Data Set," 2009, doi: 10.1109/CISDA.2009.5356528.

[3] D. H. Deshmukh, T. Ghorpade, and P. Padiya, "Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset," in *Proceedings - 2015 International Conference on Communication, Information and Computing Technology, ICCICT 2015*, Feb. 2015, pp. 1–6, doi: 10.1109/ICCICT.2015.7045674.

[4] "The Art of Feature Engineering: Essentials for Machine Learning - Pablo Duboue - Google Books." https://books.google.com/books?hl=en&lr=&id=lLbrDwAAQBAJ&oi=fnd&pg=PR11&dq=feature+engineering+machine+learning&ots=4kkpG9C7Dg&sig=fqj_G-dEKWbGNnWqOHdbsjO-MM0#v=onepage&q&f=false (accessed Mar. 01, 2021).

[5] M. C. Belavagi and B. Muniyal, "Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection," in *Procedia Computer Science*, Jan. 2016, vol. 89, pp. 117–123, doi: 10.1016/j.procs.2016.06.016.

[6] Z. Li, A. L. G. Rios, G. Xu and L. Trajković, "Machine Learning Techniques for Classifying Network Anomalies and Intrusions," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1-5, doi: 10.1109/ISCAS.2019.8702583.

[7] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *International Conference on Signal Processing and Communication Engineering Systems - Proceedings of SPACES 2015, in Association with IEEE*, Mar. 2015, pp. 92–96, doi: 10.1109/SPACES.2015.7058223.

[8] K. Bajaj, A. A. Chitkara, and H. Pradesh, "Improving the Intrusion Detection using Discriminative Machine Learning Approach and Improve the Time Complexity by Data Mining Feature Selection Methods," in *International Journal of Computer Applications* 2013, pp. 5-11, doi: 10.5120/13209-0587.

[9] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Performance Comparison of Convolutional AutoEncoders, Generative Adversarial Networks and Super-Resolution for Image Compression." in *CVPR Workshop and Challenge On Learned Image Compression*, 2018.

[10] C. C. Tan and C. Eswaran, "Using autoencoders for mammogram compression," *Journal of Medical Systems*, vol. 35, no. 1, pp. 49–58, Feb. 2011, doi: 10.1007/s10916-009-9340-3.

[11] M. Gharib, B. Mohammadi, S. H. Dastgerdi, and M. Sabokrou, "AutoIDS: Auto-encoder Based Method for Intrusion Detection System," in *arXiv*, Nov. 2019.

[12] N. O. F. Elssied, O. Ibrahim, and A. H. Osman, "A novel feature selection based on one-way ANOVA F-test for e-mail spam classification," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 3, pp. 625–638, 2014, doi: 10.19026/rjaset.7.299.

[13] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 16–28, Jan. 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[14] L. A. Álvarez Almeida and J. Carlos Martinez Santos, "Evaluating Features Selection on NSL-KDD Data-Set to Train a Support Vector Machine-Based Intrusion Detection System," Jun. 2019, doi: 10.1109/ColCACI.2019.8781803.

[15] "KDD Cup 1999 Data." http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed Mar. 01, 2021).

[16] "A Deeper Dive into the NSL-KDD Data Set | by Gerry Saporito | Towards Data Science." https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657 (accessed Mar. 01, 2021).

[17] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers and Security*, vol. 86. Elsevier Ltd, pp. 147–167, Sep. 01, 2019, doi: 10.1016/j.cose.2019.06.005.

[18] "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms | Semantic Scholar." https://www.semanticscholar.org/paper/A-Study-on-NSL-KDD-Dataset-for-Intrusion-Detection-Dhanabal-Shantharajah/f00d9a1533b296a878656db4589246a0e1597db0 (accessed Mar. 19, 2021).

[19] "Stack Overflow Developer Survey 2020." https://insights.stackoverflow.com/survey/2020#most-popular-technologies (accessed Mar. 19, 2021).

[20] "Spyder — Anaconda documentation." https://docs.anaconda.com/anaconda/user-guide/tasks/integration/spyder/ (accessed Mar. 19, 2021).

[21] D. Sarkar, R. Bali, and T. Sharma, *Practical Machine Learning with Python*. Berkeley, CA: Apress, 2018.

[22] W. Mckinney, "pandas: a Foundational Python Library for Data Analysis and Statistics." http://pandas.sf.net. (accessed: Mar. 19, 2021).

[23] "Dropout layer." https://keras.io/api/layers/regularization_layers/dropout/ (accessed Mar. 19, 2021).

[24] “Precision and Recall Definition | DeepAI.” https://deepai.org/machine-learning-glossary-and-terms/precision-and-recall (accessed Mar. 19, 2021).

[25] “Deep Learning.” https://www.deeplearningbook.org/ (accessed Mar. 19, 2021).

[26] “base/protocols/conn/main.zeek — Book of Zeek (v4.0.0).” https://docs.zeek.org/en/lts/scripts/base/protocols/conn/main.zeek.html (accessed Mar. 27, 2021).

[27] V. Shmatikov and M.-H. Wang, “Security Against Probe-Response Attacks in Collaborative Intrusion Detection”. 2007, pp. 129-136, doi: 10.1145/1352664.1352673.

[28] “KDD-CUP-99 Task Description.” http://kdd.ics.uci.edu/databases/kddcup99/task.html (accessed Mar. 30, 2021).

[29] J. W. Haines, R. Lippmann, D. Fried, M. Zissman and E. Tran, “1999 DARPA Intrusion Detection Evaluation: Design and Procedures,” 2001.

[30] “Autoencoders Tutorial | What are Autoencoders? | Edureka.” https://www.edureka.co/blog/autoencoders-tutorial/ (accessed Mar. 31, 2021).

[31] M. R. Parsaei, S. M. Rostami, and R. Javidan, “A Hybrid Data Mining Approach for Intrusion Detection on Imbalanced NSL-KDD Dataset,” in *International Journal of Advanced Computer Science and Applications*, 2016, doi: 10.14569/IJACSA.2016.070603.

APPENDIX. NSL-KDD FEATURE DESCRIPTION TABLE

Feature type	Feature number	Feature name	Description	Sample Data
Basic	1	Duration	Duration of the connection	0
	2	Protocol_type	Protocol used in the connection	Tcp
	3	Service	Destination network service used	ftp_data
	4	Flag	Status of connection	SF
	5	Src_bytes	Number of bytes transferred from source to destination in a single connection	491
	6	Dst_bytes	Number of bytes transferred from destination to source in a single connection	0
	7	Land	1 if source and destination IP addresses and port numbers are equal; else 0	1
	8	Wrong_fragment	Total number of wrong fragments in a connection	0
	9	Urgent	Number of urgent packets in a connection	0
Content	10	Hot	Number of ‘hot’ indicators in the content	0
	11	Num_failed_logins	Count of failed login attempts	0
	12	Logged_in	1 if successfully logged in; 0 otherwise	1
	13	Num_compromised	Number of “compromised” conditions	0
	14	Root_shell	1 if root shell is obtained; 0 otherwise	0
	15	Su_attempted	1 if ``su root'' command attempted or used; 0 otherwise.	0
	16	Num_root	Number of ``root'' accesses or number of operations performed as a root in the connection	0
	17	Num_file_creations	Number of file creation operations in the connection	0
	18	Num_shells	Number of shell prompts	0
	19	Num_access_files	Number of operations on access control files	0
	20	Num_outbound_cmds	Number of outbound commands in an ftp session	0
	21	Is_hot_login	1 if the login belongs to the “hot'' list	0
	22	Is_guest_login	1 if the login is a guest login	0

Feature type	Feature number	Feature name	Description	Sample Data
Time-related	23	Count	Number of connections to the same destination host is the current connection in the past two seconds	2
	24	Srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds	2
	25	Serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)	0
	26	Srv_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)	0
	27	Rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)	0
	28	Srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)	0
	29	Same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)	1
	30	Diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count (23)	0
	31	Srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)	0
Host-based	32	Dst_host_count	Number of connections having the same destination host IP address	150
	33	Dst_host_srv_count	Number of connections having the same port number	25
	34		The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32)	0.17
	35	Dst_host_diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)	0.03
	36	Dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)	0.17
	37	Dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)	0
	38	Dst_host_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)	0

Feature type	Feature number	Feature name	Description	Sample Data
	39	Dst_host_srv_serror_rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)	0
	40	Dst_host_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)	0.05
	41	Dst_host_srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)	0