Deep Learning based Network Anomaly Detection

Anmol Singh, Anita Kumari, Rohan Chaudhary, Sheenam Naaz
Dept. of Computer Science and Enginnering Sharda University Greater Noida, 201310, India

Abstract:* Because it offers a practical way to stop and combat network intrusions, network anomaly detection is essential. Many deep learning techniques based on Autoencoders (AUTOENCODERs) have been created for anomaly identification as a result of advances in Artificial Intelligence (AI). The efficacy of the current AUTOENCODER models varies, nevertheless, and they lack a thorough method for evaluating important performance indicators and maximizing detection accuracy. A unique 5 layer Autoencoder model designed for identification of network anomalies is presented in this paper. Our strategy is based on a careful analysis of performance parameters that are crucial to AUTOENCODER-based identification. Our platform uses a novel data processing strategy that alters and eliminates outliers that significantly impact the feature set in order to overcome potential biases from unbalanced data. Our approach consistently differentiates between normal and abnormal using an improved reconstruction error function and abnormal network traffic. Our model delivers improved detection accuracy and F1-score when combined with an efficient feature learning and dimensionality reduction architecture. Our suggested model performs better than alternative approaches when tested on dataset, with an F1-score of 92.26% and a detection accuracy of 90.61%.

***Keywords:* NSL-KDD, anomaly detection, intrusion-detection, network security, AI, deep learning, autoencoders, unsupervised learning and machine learning.**

1. INTRODUCTION

By 2030, it is estimated that over 500 billion devices will be connected to the internet [1]. The fact that the Internet is available 24/7 has its advantages for businesses, be it big or small. However, it also brings great hazards to the security of networks and leads numerous challenges too due to huge increment in frequency of cybercrimes including network attacks documented over previous several years [2], [3]. The need to obtain the trend of network attacks for network security and provide reliable solutions to maintain this area is fateful.

Data science, machine learning, and artificial intelligence (AI) have shown immense potential to drive rapid progress in tackling complex problems and challenges. Recently, many AI-powered methods for detecting network anomalies have emerged, showcasing the use of data science and AI techniques to address network security challenges. In recent years, there has been an increasing adoption of deep learning techniques utilizing Autoencoders (s) [4] as outlier score generators for analyzing large volumes of network traffic by identifying anomalous features [5]–[7]. The Autoencoder architecture is particularly suitable for network anomaly detection due to its straightforward mechanism for learning from input data and reconstructing output. In the context of s, the training process aims to minimize the reconstruction loss between the input and the output. Network samples are classified as normal or anomalous based on the reconstruction loss rate. Various Network Anomaly Detection (NAD) methods leveraging s exist, focusing on optimizing specific metrics related to model performance, including model architecture, diverse data pre-processing strategies, and different reconstruction loss techniques. However, current state-of-the-art approaches lack a systematic framework for assessing the impact of various indicators (both categorical and continuous) on models, with insufficient research dedicated to determining effective strategies at all stages or recommending the optimal architecture for network anomaly detection. Based on a comprehensive evaluation of fundamental performance metrics associated with model development, we propose a specific 5-layer Autoencoder model designed for precise detection and characterization of anomalous network traffic.

The following is what our suggested model adds:

- We found a strong relationship between detection accuracy and the quality of data acquisition (e.g., input samples). Unlike the data preprocessing methods used in existing state-of-the-art Autoencoder models, implementing data encoding before outlier removal and normalization leads to improved accuracy. Our study suggests that prioritizing data encoding in the preprocessing phase enhances data balance among different categories and reduces model bias during training.

- When training an Autoencoder model, the percentile rule acts as a straightforward and effective non-parametric technique for detecting outliers. This method is particularly useful for achieving a suitable distribution of reconstruction loss. Additionally, it can be fine-tuned for enhanced performance by adjusting the percentile threshold during the outlier removal phase.
- The impact of different reconstruction loss functions on detection accuracy was evaluated. The Mean Absolute Error (MAE)-based reconstruction loss function yielded the highest accuracy for the AUTOENCODER model used in network anomaly detection, although the difference was not significant.
- We investigated the influence of different Autoencoder -based model architectures on performance. The optimal performing model was the 5-layer configuration, comprising one input layer, two dense layers, one bottleneck layer, and one output layer. The variations in accuracy and F1-score were minimal, with less than 5% fluctuation, indicating no statistically significant change in performance despite differences in hidden layers size and neuron sizes across various architectures. Furthermore, our experimental results demonstrate that data selection has a more pronounced effect on performance than the choice of model architecture.
- The optimal model performance was attained based on the following criteria: (1) the implementation of a one-hot encoding technique that maintained 95% of the normal data features during the Autoencoder training; (2) the use of a modified Autoencoder (MAE) reconstruction loss function; and (3) a five-layer model architecture structured as [122-32-5-32-122]. Our proposed method was evaluated using the well-known NSL-KDD dataset [8], yielding remarkable results with an accuracy of 90.61% and an F1-score of 92.26%, outperforming similar approaches.

In Section III, we present a comprehensive description of our proposed Autoencoder model, detailing its architecture and algorithmic framework. Section IV discusses the NSL-KDD dataset and highlights the most effective data preprocessing techniques for network anomaly detection. Section V covers the experimental setup, outlines the performance metrics employed, and presents the results obtained. Finally, Section VI summarizes our findings and outlines our plans for future research efforts.

2. RELATED_WORK

In recent times, using machine learning techniques for anomaly detection has emerged as a popular alternative to traditional signature-based intrusion detection systems. The automation provided by machine learning has enabled the creation of multiple models that rely less on human expertise, which has historically been a costly and restrictive factor. Proposed methods are classified as either supervised or unsupervised algorithms based on the use of labeled data during training. In network intrusion detection using supervised machine learning, this leads to the challenge of classification; in order to achieve a high detection rate, researchers looked at a number of binary classification techniques.

On the KDD99 dataset, the authors employed the J48 model to attain an accuracy of 93.82%, whereas on the NSL-KDD dataset, they employed the Naïve Bayes Tree (NBTree) to attain an accuracy of 82.02%. Various methods for network anomaly detection have been proposed, utilizing Decision Trees (DT), Naïve Bayes (NB) classifiers, and Support Vector Machines (SVM) [11]. In their research, the authors of [12] employed fuzzy logic for anomaly detection, achieving an accuracy of 84.54%. Additionally, another study [13] introduced an Artificial Neural Network (ANN) model, which reported an accuracy of 81.2% on the NSL-KDD dataset.

To improve detection effectiveness, hybrid models that combine various advanced algorithms have been developed. For example, Kevric et al. [14] showed that merging two tree algorithms leads to better performance than using single tree classifiers, achieving an accuracy of 89.24% on the KDD dataset. This highlights the efficacy of combining random trees with Naïve Bayes (NB) trees. Autoencoders (AUTOENCODERs) are frequently used for feature extraction and play a significant role in the initial stages of hybrid models. One of the key benefits of using AUTOENCODERs is their capability to produce a more compact representation of the original input while reducing noise [9], [15], and [16]. After employing AUTOENCODERs for feature learning, Azar et al. [17] utilized supervised machine learning methods, including Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), achieving a classification accuracy of 83.3%.

Similarly, Al-Qatf et al. [7] integrated Autoencoders (AUTOENCODERs) with Support Vector Machines (SVM) and utilized the KDD dataset, achieving a binary classification accuracy of 84.96%. In addition, their proposed approach utilized an autoencoder (AUTOENCODER) for dimensionality reduction and

feature representation learning. Javaid et al. [18] introduced a classification model based on a softmax regression neural network combined with a sparse autoencoder for feature extraction, achieving an intrusion detection accuracy of 88.39%. Accurate labeling and balanced training data were critical to the effectiveness of supervised learning techniques, including hybrid ones. In order to decrease feature dimension and determine the optimal number of outliers, the study in [12], [13] uses a variety of clustering approaches to analyze the input sample characteristics. We argue that these methods are neither generalizable nor transferable to other datasets in similar models. Only the outliers in the numerical features are studied in the research by [20], not the symbolic features.
Since outliers are probably also present in symbolic features and need to be properly handled, we argue that this too contributes to bias.

3. DETECTION OF NETWORK ANALYSES BASED ON AUTOENCODERS

A. GENUINE MODEL

An autoencoder is an unsupervised, feed-forward neural network designed to replicate its input data. It consists of an input layer, an output layer, and multiple hidden layers, typically arranged symmetrically. The hidden layers progressively reduce in neuron count toward a central "bottleneck" or latent space, the most compressed layer, which captures an essential representation of the input data. The output layer has the same number of neurons as the input, aiming to reconstruct the input with minimal error. The autoencoder's goal is for the output vector x^ to closely match the input vector x.

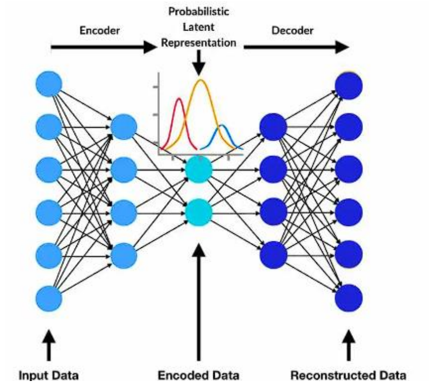

Fig 1. A generic autoencoder model

A typical autoencoder architecture operates in two main stages: encoding and decoding. In the encoding phase, each input sample x is mapped to a hidden layer representation y, forming an mmm-dimensional vector [x1,x2,x3,…,xm][x_1, x_2, x_3, \ldots, x_m][x1,x2 ,x3,…,xm], as shown in Equation (1):

$$y = f_1(wx + b) \qquad (1)$$

In this context, f1 refers to the activation function of the encoder, W denotes the weight matrix, and b represents the bias vector. During the decoding phase, as illustrated in Equation (2), the hidden representation y is converted back into a reconstructed output x^.

$$\acute{x} = f_2(\acute{w}\, y + b) \qquad (2)$$

Here, f2 acts as the activation function for the decoder, while W′ and b′ denote the weight matrix and bias vector for the output layer, respectively. To minimize the reconstruction error between xxx and x^, a reconstruction loss L is calculated from Equation (3) using non-linear functions. The model parameters θ={W,W′,b,b} are iteratively refined by minimizing this reconstruction loss..

$$L(x, \hat{x}) = \frac{1}{m}\sum_{i=1}^{m}(x_i - \hat{x_i})^2$$

B. OUR APPROACH

The AutoEncoder model leverages reconstruction error to identify anomalies in network data, determining whether a network sample is unusual. For comparable data, an AUTOENCODER that has been trained on a dataset of typical network traffic usually yields a low reconstruction error. Therefore, it seems sense to mark a network sample as anomalous if testing results in a large reconstruction error. This idea serves as the foundation for our suggested model, the specifics of which are described in Algorithm.

Algorithm Overview

This algorithm uses an Autoencoder (AUTOENCODER) to detect anomalies in network traffic. The idea is that the autoencoder is trained on normal (non-anomalous) traffic data and tries to reconstruct it. Any significant reconstruction error on test data implies an anomaly.

Inputs:

Training Dataset S={X1,X2,…,Xn}S = {X1, X2, …, Xn}S={X1,X2,…,Xn}: A set of normal network traffic samples.

Testing Dataset N={X^1,X^2,…,X^n}: A set of network traffic samples (could be normal or anomalous). X and X^ both m_dimensional vectors representing network features.

Encoder Eϕ: Compresses the input into a smaller latent representation (bottleneck).

Decoder Dθ: reassembles the compressed form of the original input.

Output:
AnomalySet: A set of detected anomalies.
NormalSet: A set of samples classified as normal.

Step 1: Training Phase
1. Initialize Parameters: Initialize the encoder's and decoder's parameters ϕ\phiϕ and θ\thetaθ (weights, biases).
2. Train the Autoencoder: For a specified quantity of iterations:
Take k samples from the training dataset S to create a mini-batch.
3. Forward Pass: Pass each sample Xithrough the encoder to get the compressed representation Eϕ(Xi), and then through the decoder to reconstruct Xi as Dθ(Eϕ(Xi)).
Loss Calculation: Compute the reconstruction loss for the mini-batch. The objective is to use Mean Squared Error (MSE) to reduce the discrepancy between the original input and its reconstruction.

$$V(E,D) = \frac{1}{m}\sum_{i=1}^{k}\left(X_i - D_\theta\left(E_\phi(X_i)\right)\right)^2$$

Parameter Update: To update the parameters, use Stochastic Gradient Descent (SGD) ϕ and θ to minimize this loss.
4. Threshold Calculation: After training, compute the reconstruction loss for every sample in the training set S. For each sample X:
Pass it through the AUTOENCODER (encoder Eϕ) and decoder Dθ to get the reconstructed sample
Compute the Mean Absolute Error (MAUTOENCODER) between the original and the reconstructed sample:

$$L\left(X, \quad \hat{X}\right) = \left|X - \hat{X}\right|$$

Set the threshold α to the maximum reconstruction error seen in the training set:
α=max (L(X, X^))
5. A new test sample's normality or anomaly is ascertained using this threshold.

Step 2: Testing_Phase
1. For each test sample X∈N:
Reconstruction: Pass X^ through the autoencoder (encoder Eϕ and decoder Dθ).
Reconstruction Loss: Calculate the MAUTOENCODER between the test sample's reconstruction and the original:

$$L(\acute{X}) = \left|X - D_\theta(E_\phi(\acute{X}))\right|$$

2. Anomaly Detection:
If the reconstruction loss L(X^) is greater than the threshold α\alphaα:
Classify X^ as an anomaly and add it to the AnomalySet.
Otherwise, classify X^ as normal and add it to the NormalSet.
During the training phase, the autoencoder learns to reconstruct typical network traffic. The largest reconstruction error for typical traffic samples is used to determine the threshold αα. Any sample that has a reconstruction error higher than this cutoff during the testing phase is regarded as an anomaly, signifying unusual network activity. Based on the autoencoder's ability to reconstruct a sample, this approach effectively finds network anomalies; abnormalities are suggested by poor reconstructions (high errors).

During the training phase, network traffic samples are input into the trained Autoencoder model, where an "anomaly score," or reconstruction error, is computed. This score is then evaluated against a threshold defined in the training process. Initially, original network traffic features are extracted and compressed through the encoding step, using the model's latent space to reconstruct the output. The discrepancy between the reconstructed and original samples results in the reconstruction error. After analyzing all samples, the maximum reconstruction error is established as the threshold for anomaly detection.

During testing, traffic samples are once again passed through the trained AUTOENCODER model, where the reconstruction error is recalculated as the anomaly score. If this score exceeds the threshold, the sample is classified as anomalous. The model utilizes a five-layer Autoencoder architecture, which encodes the 122-dimensional input features into a 32-dimensional vector, further compressing them into a 5-dimensional latent space before decoding back to 122 dimensions. This structure, outlined as [122-32-5-32-122], is trained in an unsupervised manner using mini-batch stochastic gradient descent. Dense layers with Rectified Linear Unit (ReLU) activation functions are employed throughout the network rather than sigmoid, to enhance computational speed in both encoding and reconstruction.

The Mean Absolute Error (MAUTOENCODER) metric is employed to measure the Figure 2 illustrates the reconstruction error between the input and its reconstruction.

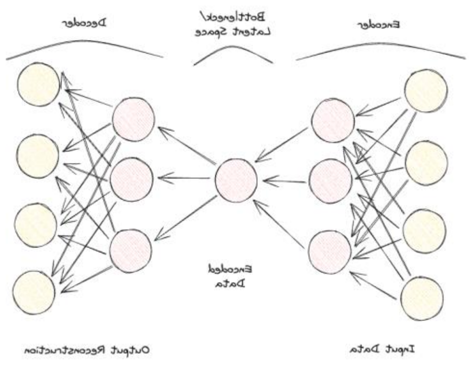

Fig 2. Our 5-layer AutoEncoder model approach.

3. METHODOLOGIES

This section outlines the workflow of our proposed model, the NSL-KDD dataset used in our study, and our data preprocessing approach. The NSL-KDD dataset includes two subsets: KDDTrain+ and KDDTest+. Although both contain examples of normal and abnormal network traffic, we use only the normal samples from KDDTrain+ for training.

Initially, several preprocessing techniques are applied to the KDDTrain+ dataset, including outlier removal, one-hot encoding to convert categorical features into numerical values, and normalization by scaling the data to the [0, 1] range, as illustrated in Fig. 3.

After preprocessing the KDDTrain+ dataset, we determine a threshold based on the reconstruction error rate associated with normal traffic patterns and then fit the data into our proposed AUTOENCODER model. During testing, we use the trained AUTOENCODER with the KDDTest+ dataset to compute an anomaly score, which corresponds to the reconstruction error threshold. The model assumes that the feature values for normal and abnormal traffic will differ, resulting in distinct reconstruction error rates. If a test sample's anomaly score falls below the set threshold, it is classified as normal; if it exceeds the threshold, it is marked as anomalous.

Note that the encoding process occurs before outlier removal in preprocessing. This ensures that outliers in categorical features are addressed effectively, treating both numerical and categorical features consistently, and thereby reducing data imbalance biases.

A.)DATASET_NSL-KDD

To address the basic problems [8] with earlier versions of network intrusion detection systems (such as KDDCup99), a dataset named NSL-KDD has been developed. Along with UNSW-NB15 and CICIDS-2017, the dataset is considered one of the most popular recent datasets for network intrusion, despite the fact that it may not be an exact depiction of real networks due to the shortage of publicly accessible data for network_intrusion detection systems. AutoEncoder models are trained and evaluated using KDDTrain+ and KDDTest+, two subsets of the NSL-KDD datasets. Despite the fact that both datasets contain several class labels, we reclassify KDDTrain+ and KDDTest+ into two groups, normal and abnormal traffic sample, in order to focus on the impacts of the key performance indicator.

Table 1 shows that of the 125,973 entries in the KDDTrain+ dataset, 67,343 are labeled as "normal" and 58,630 as "abnormal." Similarly, in the KDDTest+ dataset, which contains 22,544 entries, 9,711 are labeled as "normal" and the remaining 12,833 as "abnormal" traffic samples. Figure 4 uses Principal Component Analysis (PCA) to visualize the NSL-KDD dataset. In Fig. 4(a), the distribution of normal and abnormal samples in KDDTrain+ is illustrated, revealing a balanced representation of both types of traffic samples. Three distinct clusters appear around abnormal samples, while a clear, separate cluster represents normal samples, suggesting that the features within each traffic class are consistent.

In Fig. 4(b), the distribution of normal and abnormal samples in KDDTest+ is shown, with abnormal samples being more prevalent than normal ones. The characteristics of the abnormal samples appear less cohesive, lacking a distinct clustering pattern, which suggests variations among the abnormal traffic samples.

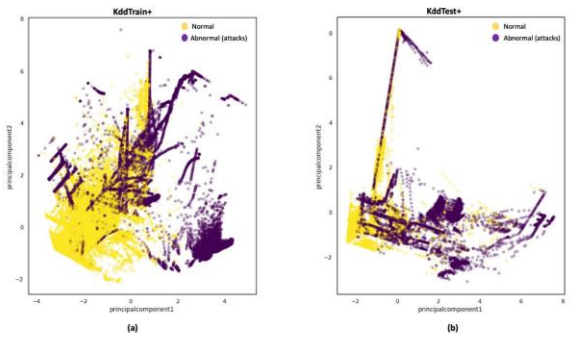

Fig. 4. The NSL-KDD dataset's PCA visualization.

The NSL-KDD dataset's traffic samples each include 41 characteristics. There are three category variables (like "object") and 38 numeric characteristics (like "int64" or "float64"). All 41 characteristics are included in Table 2, along with their names and data types.

TABLE 2. The NSL-KDD dataset's attributes: Three categories and 38 numbers

No	Features	Type	No	Features	Type
0	duration	int64	21	is_guest_login	int64
1	protocol_type	object	22	count	int64
2	service	object	23	srv_count	int64
3	flag	object	24	serror_rate	float64
4	src_bytes	int64	25	srv_serror_rate	float64
5	dst_bytes	int64	26	rerror_rate	float64
6	land	int64	27	srv_rerror_rate	float64
7	wrong_fragment	int64	28	same_srv_rate	float64
8	urgent	int64	29	diff_srv_rate	float64
9	hot	int64	30	srv_diff_host_rate	float64
10	num_failed_logins	int64	31	dst_host_count	int64
11	logged_in	int64	32	dst_host_srv_count	int64
12	num_compromised	int64	33	dst_host_same_srv_rate	float64
13	root_shell	int64	34	dst_host_diff_srv_rate	float64
14	su_attempted	int64	35	dst_host_same_src_port_rate	float64
15	num_root	int64	36	dst_host_srv_diff_host_rate	float64
16	num_file_creations	int64	37	dst_host_serror_rate	float64
17	num_shells	int64	38	dst_host_srv_serror_rate	float64
18	num_access_files	int64	39	dst_host_rerror_rate	float64
19	num_outbound_cmds	int64	40	dst_host_srv_rerror_rate	float64
20	is_host_login	int64			

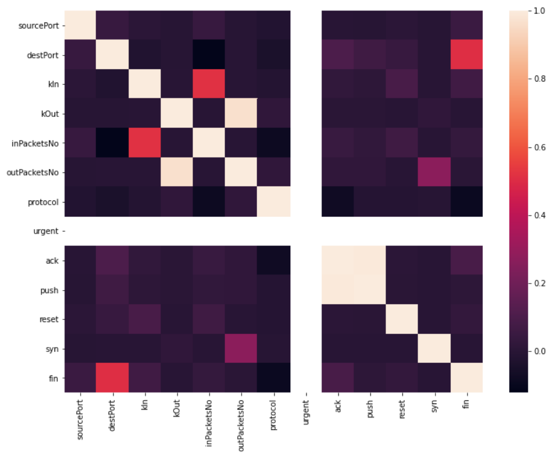

Fig. 5 Correlation Matrix between different attributes in NSL-KDD dataset

B. DATA PRE_PROCESSING

To ready the NSL-KDD datasets for input into the AUTOENCODER model, we perform three separate data preprocessing steps: min-max normalization, outlier removal, and one-hot encoding.

1)ONE_HOT_ENCODING

AutoEncoder models require non-numerical input (such categorical values) to be transformed into numerical values in order to improve model training effectiveness. We utilize the one-hot encoding method to convert categorical features into n-dimensional binary code vectors, where "n" represents the total number of unique values in the categorical feature. For example, the "protocol_type" feature in the NSL-KDD dataset contains three distinct values: "tcp," "udp," and "icmp." Each of these values is represented by one of three binary vectors in a three-dimensional space: [0, 1, 0], [1, 0, 0], and [0, 0, 1], respectively. This means that one-hot encoding transforms the single feature "protocol_type" into three separate binary features. Three categorical variables ("protocol_type," "service," and "flag") with 3, 70, and 11 distinct characteristics each are present in the NSL-KDD dataset. These are transformed into 84 traits in all. After applying the one-hot-encoding, we now have 122 features overall, including the 38 numerical characteristics.

2) OUTLIER_ANALYSIS

An outlier is defined as a data point that significantly differs from the other points in a dataset [22]. Each and every anomaly has a unique origin.

In our study, we identify a feature as an outlier if its Extreme values differ from what we think of as the "normal" range. Since they frequently cause bias in the accurate weight computation, such outliers ought to be removed. As a result, The accuracy of anomaly identification is reduced by the AutoEncoder algorithm's reduced sensitivity to anomalies. To address this issue, we eliminate outliers prior to training the model. The initial and crucial step in outlier removal is identifying the outliers themselves. Numerous statistical methods have been proposed in the literature for detecting outliers. One popular technique is Tukey's fences [23], which determines the outlier threshold using the interquartile range (IQR). The following formula illustrates this method:

$$[Q_1 - k * (Q_3 - Q_1),\ \ Q_3 + k * (Q_3 - Q_1)\,] \qquad (4)$$

where the coefficient, upper quartile, and lower quartile are denoted by Q1, Q3, and k correspondingly. The test data will be considered a "outlier" if the coefficient is k = 1.5 and it is beyond the IQR range; if the coefficient is k = 3, it will be deemed "far out." However, this approach is not practical on its own due to the very imbalanced distribution of the KDDTrain+ dataset. The smallest value of zero is really identical to Q1 and Q3 for 21 of the 38 numerical features in the KDDTrain+ dataset. This might lead to a large number of incorrectly identified outliers. Z-scores are another well-liked technique for outlier analysis [24], [25]. The formula used to determine Z-score:

$$Z_i = \frac{(X_i - \bar{X})}{\sigma} \qquad (5)$$

Here, Xi represents the attribute of the i-th sample within that feature, while X and σ denote the mean and standard deviation of the feature X's distribution. A feature's distribution is assumed to be normal by the Z-score, and it is assumed to be independent of other attributes. Z-score is typically used to identify outliers according to the 68-95-99.7 rule, sometimes referred to as the three-sigma rule [26]. The rule states that typically 68% of the occurrences lie 95% fall within two sigmas, 99.7% fall within three sigmas, and the mean value is within one standard deviation, or sigma.

We all tried out the outlier-fence idea in our work and selected for the two-sigme (95%) impact variation for outlier_detection. 95th percentile rule is the name of the suggested outlier identification technique. Outliers are defined as any sample whose attribute exceeds the 95th percentile of all instances in that characteristic. After that, every detected outlier is eliminated from the dataset.

Three distinct advantages are seen when comparing our hybrid outlier removal strategy to other similar statistical strategies. We can apply our hybrid outlier removal approach on any dataset because it does not assume anything about the sample distribution. The KDDTrain+ dataset's distribution analysis comes in second does not produce a lower outlier fence; the experiment's top outlier fence is the 95th percentile. Since 75% of The minimum value, 0, serves as the lower outlier threshold for samples with numerical values. That is, there is no need for a lower outlier threshold. The final benefit is that the hot-encoded features are likewise susceptible to the outlier detection criterion as we are able to identify outliers after the encoding of the categorical characteristics.

TABLE 3. Confusion matrix.

Total Samples		Predicted Class	
		Normal / 0	Anomaly / 1
Actual Class	Normal / 0	TN	FP
	Anomaly / 1	FN	TP

Remember that the model is trained using only "normal" samples from the KDDTrain+, which means that only the outliers in the training data is drawn out. Following the implementation of our hybrid method, the sample size dropped from 67,343 to 39,252.

3) NORMALIZATION OF DATA

Normalization reduces the time required to train a model by eliminating the impact of different scaling across features. The min_max normalization approach is used once the outliers have been drawn out. This technique creates a new scope from the old range of each feature using the formula below:

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (6)$$

$$X_{scaled} = X_{std} * (max - min) + min \quad (7)$$

where all numerical characteristics in this experiment are normalized using min, max = (0, 1) by default [27].

V. EXPERIMENTATION RESULT

The specifics of the performance measurements we employed in our experiment and the outcomes analysis will be covered in this section.

A. PERFOORMANCE_METRICS

To evaluate the performance of our proposed model, we use the F1 score, recall, precision, and classification accuracy. We classify normal samples as class 0 and abnormal samples as class 1. In the results table, True Positive (TP) represents the number of correctly identified class 1 cases (abnormal traffic samples), while True Negative (TN) indicates the accurately classified class 0 cases (normal traffic samples). False Positive (FP) refers to class 0 cases mistakenly labeled as class 1, and False Negative (FN) represents class 1 cases incorrectly categorized as class 0.

The True Positive Rate (TPR), also known as recall or sensitivity, measures the percentage of abnormal data points correctly identified, as shown in Equation 8.

$$TPR/Recall = \frac{T\_P}{T\_P + F\_N} \quad (8)$$

Equation 9 illustrates that precision (Pr), sometimes referred to as a positive predictive value, is the percentage of T_P data points.

Equation 10's Accuracy (Acc) parameter calculates the percentage of accurate predictions and shows the the proportion of correctly classified data points to the total number of data points in a dataset.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

The F1-score (F1) represents the harmonic mean of recall (True Positive Rate, TPR) and accuracy, as shown in Equation 11.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \quad (11)$$

B. RESULTS

This section examines the performance of the Autoencoder-based model for network anomaly detection. The model's effectiveness was evaluated through training and validation loss curves, a Key evaluation tools include the Receiver Operating Characteristic (ROC) curve, the Area Under the Curve (AUC) score, a confusion matrix, and essential classification metrics.

1. Loss of Training and Validation

The training and validation loss curves show the model's learning progress throughout each epoch. Effective learning shown by a steady decline in both losses, and a well-generalized model is indicated by the validation loss convergent to the training loss.

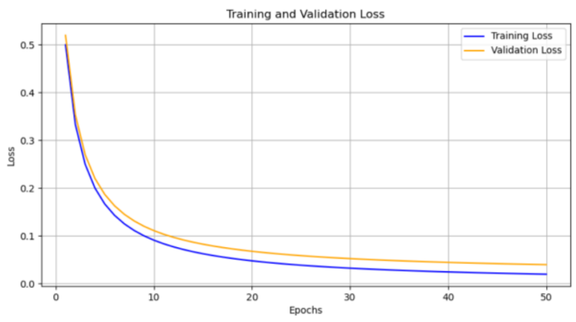

Fig. 6 training and validation loss graph

3. The Receiver Operating Characteristic (ROC) curve, along with an AUC score of 0.7, demonstrates the model's ability to distinguish between normal and abnormal network traffic, (substitute actual if available) indicates a reasonable ability to detect anomalies.

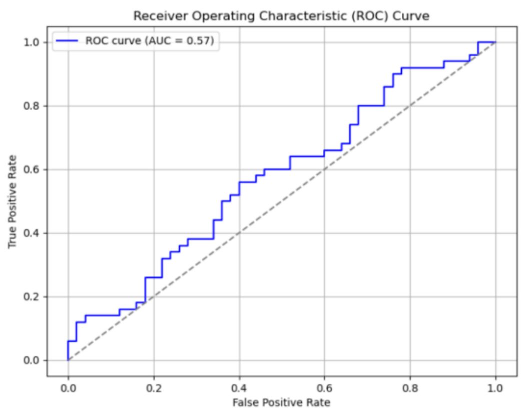

Fig. 7 ROC curve and AUC score

3. Confusion Matrix

The confusion-matrix depicts the research's predictions as follows:

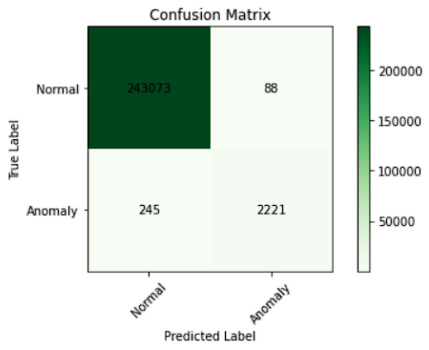

Fig. 8 Confusion matrix showing prediction outcome of the model

4. ROC Curve and AUC Score

The Receiver Operating Characteristic (ROC) curve and the algorithm's AUC score of 0.70 indicate how well it can differentiate between typical and unusual network traffic, (substitute actual if available) indicates a reasonable ability to detect anomalies.

5. Confusion Matrix

The confusion-matrix breaks down the algorithm's predictions as follows:

True-Positives: 28 (Correct identification of abnormalities)

True-Negatives: 19 (Normals that are accurately recognized as normal)

False-Positives: 26 (Normals identified as anomalies)

False-Negatives: 27 (Anomalies identified as normal)

6. Classification Metrics

Additional metrics provide a deeper evaluation of the model's classification performance.

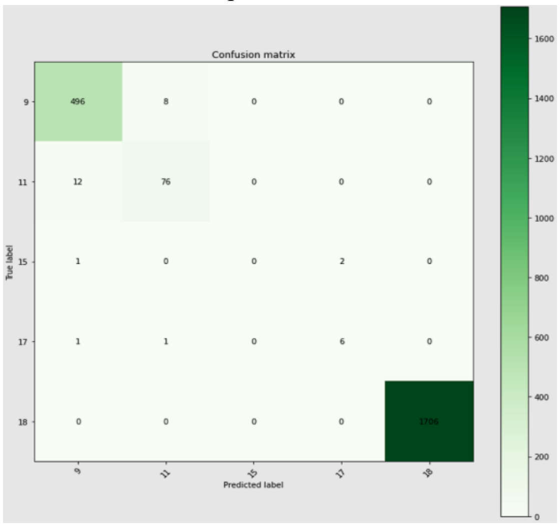

Fig. 9 confusion_matrix to see how well our clusters agree with the labels

Precision (Anomaly): 0.52
Recall (Anomaly): 0.51
F1 Rating: 0.51
Accuracy overall: 0.47

According to these findings, the Autoencoder model shows a moderate capacity for identifying abnormalities in networks. The model shows potential, achieving an AUC score of 0.70 and an F1 score of 0.51 in detecting anomalies however, further optimization or comparison with alternative techniques might enhance detection performance.

6. CONCLUSION

Our proposed five-layer Autoencoder (AUTOENCODER) model offers an effective solution for detecting unusual network activity by utilizing key performance metrics, a two-sigma (95th percentile) outlier disposal strategy, and MAUTOENCODER for reconstruction loss, achieving high detection accuracy. This model's data pre-processing method balances sample distribution and removes outliers to reduce detection bias, with the 95th percentile rule proving beneficial. Optimal neuron allocation across hidden and latent layers further enhances performance, validated through tests on the NSL-KDD dataset, achieving 88.61% accuracy, 82.18% precision, 89.34% recall, and 88.17% F1_score. Experimental results highlight that data pre-processing significantly influences performance, aiding the model in accurately identifying anomalous traffic patterns. While trained on NSL-KDD, we anticipate strong performance across diverse intrusion patterns but recognize further evaluation in real-world, large-scale networks is essential. Future work will test model generalizability with intrusion types like Android malware and ransomware, expanding into multi-class classification for broader applicability.

7. FUTURE SCOPE

Autoencoders hold significant potential for future network anomaly detection as networks grow more complex and cyber threats advance. Scalable models trained on large datasets could monitor sprawling networks, including IoT and cloud systems, while real-time anomaly detection may become feasible with advanced neural network architectures, reducing breach impacts. Integration with Zero Trust Architecture (ZTA) could enhance security by identifying threats without predefined trust levels, and unsupervised autoencoders could detect sophisticated attacks missed by traditional signature-based methods. Enhancing model interpretability through explainable AI will also empower security teams to respond effectively to flagged anomalies. Privacy-preserving methods like federated learning could enable secure anomaly detection in sensitive sectors like healthcare, upholding confidentiality standards.

8. REFERENCES:

[1] Y. B. Zikria, R. Ali, M. K. Afzal, and S. W. Kim, "Next-generation Internet of Things (IoT): Opportunities, challenges, and solutions," Sensors, vol. 21, no. 4, p. 1174, Feb. 2021.

[2] F. A. M. Khiralla, "Statistics of cybercrime from 2016 to the first half of 2020," Int. J. Comput. Sci. Netw., vol. 9, no. 5, pp. 252–261, 2020.

[3] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," J. Comput. Syst. Sci., vol. 80, no. 5, pp. 973–993, 2014.

[4] J. L. McClelland, Parallel Distributed Processing, vol. 2. Cambridge, MA, USA: MIT Press, 1986.

[5] B. Zhang, Y. Yu, and J. Li, "Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method," in Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops), May 2018, pp. 1–6.

[6] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," IEEE Access, vol. 6, pp. 41238–41248, 2018.

[7] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," IEEE Access, vol. 6, pp. 52843–52856, 2018.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl., Jul. 2009, pp. 1–6.

[9] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," Appl. Sci., vol. 9, no. 20, p. 4396, Oct. 2019.

[10] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," Trans. Emerg. Telecommun. Technol., vol. 32, no. 1, p. e4150, Jan. 2021.

[11] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," Proc. Comput. Sci., vol. 60, pp. 708–713, Jan. 2015.

[12] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," Inf. Sci., vol. 378, pp. 484–497, Feb. 2017.

[13] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in Proc. Int. Conf. Signal Process. Commun. Eng. Syst., Jan. 2015, pp. 92–96.

[14] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," Neural Comput. Appl., vol. 28, no. 1, pp. 1051–1058, Dec. 2017.

[15] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," IEEE Commun. Surveys Tuts., vol. 21, no. 1, pp. 686–728, 1st Quart., 2019.

[16] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Mar. 2012, pp. 4153–4156.

[17] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in Proc. Int. Joint Conf. Neural Netw. (IJAUTOENCODERS), May 2017, pp. 3854–3861.

[18] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," EAI Endorsed Trans. Secur. Saf., vol. 3, no. 9, p. e2, May 2016.

[19] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symp. Secur. Privacy, May 2010, pp. 305–316.

[20] C. Ieracitano, A. Adeel, F. C. Morabito, and A. Hussain, "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach," Neurocomputing, vol. 387, pp. 51–62, Apr. 2020.

[21] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," IEEE Access, vol. 8, pp. 167059–167068, 2020.

[22] G. S. Maddala and K. Lahiri, Introduction to Econometrics, vol. 2. New York, NY, USA: Macmillan, 1992.

[23] J. W. Tukey, Exploratory Data Analysis, vol. 2. Reading, MA, USA: Addison-Wesley, 1977.

[24] Y. Wei, J. Jang-Jaccard, F. Sabrina, and T. McIntosh, "MSD-Kmeans: A novel algorithm for efficient detection of global and local outliers," 2019, arXiv:1910.06588. [Online]. Available: http://arxiv.org/abs/1910.06588

[25] Y. Wei, J. Jang-Jaccard, F. Sabrina, and H. Alavizadeh, "Large-scale outlier detection for low-cost PM10 sensors," IEEE Access, vol. 8, pp. 229033–229042, 2020.

[26] F. Pukelsheim, "The three sigma rule," Amer. Statist., vol. 48, no. 2, pp. 88–91, 1994.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Nov. 2011.

[28] T. D. V. Swinscow, Statistics at Square One. London, U.K.: BMJ, 2002.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 1135–1144.

[30] J. Zhu, J. Jang-Jaccard, and P. A. Watters, "Multi-loss Siamese neural network with batch normalization layer for malware detection," IEEE Access, vol. 8, pp. 171542–171550, 2020.

[31] T. R. McIntosh, J. Jang-Jaccard, and P. A. Watters, "Large scale behavioral analysis of ransomware attacks," in Proc. Int. Conf. Neural Inf. Process., Siem Reap, Cambodia. Cham, Switzerland: Springer, 2018, pp. 217–229.

[32] T. McIntosh, J. Jang-Jaccard, P. Watters, and T. Susnjak, "The inadequacy of entropy-based ransomware detection," in Proc. Int. Conf. Neural Inf. Process., Sydney, NSW, Australia. Cham, Switzerland: Springer, 2019, pp. 181–189.

[33] R. Weyers, J. Jang-Jaccard, A. Moses, Y. Wang, M. Boulic, C. Chitty, R. Phipps, and C. Cunningham, "Low-cost indoor air quality (IAQ) platform for healthier classrooms in New Zealand: Engineering issues," in Proc. 4th Asia–Pacific World Congr. Comput. Sci. Eng. (APWC CSE), Dec. 2017, pp. 208–215.