IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 9, 2021, accepted September 27, 2021, date of publication September 29, 2021, date of current version October 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3116612

Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset

WEN XU[1], JULIAN JANG-JACCARD[1], AMARDEEP SINGH[1], YUANYUAN WEI[1], AND FARIZA SABRINA[2], (Member, IEEE)
[1]Cybersecurity Laboratory, Massey University, Auckland 0632, New Zealand
[2]School of Engineering and Technology, Central Queensland University, Sydney, NSW 2000, Australia

Corresponding author: Wen Xu (w.xu2@massey.ac.nz)

This work was supported by the Cyber Security Research Programme—Artificial Intelligence for Automating Response to Threats from the Ministry of Business, Innovation, and Employment (MBIE) of New Zealand as a part of the Catalyst Strategy Funds under Grant MAUX1912.

ABSTRACT Network anomaly detection plays a crucial role as it provides an effective mechanism to block or stop cyberattacks. With the recent advancement of Artificial Intelligence (AI), there has been a number of Autoencoder (AE) based deep learning approaches for network anomaly detection to improve our posture towards network security. The performance of existing state-of-the-art AE models used for network anomaly detection varies without offering a holistic approach to understand the critical impacts of the core set of important performance indicators of AE models and the detection accuracy. In this study, we propose a novel 5-layer autoencoder (AE)-based model better suited for network anomaly detection tasks. Our proposal is based on the results we obtained through an extensive and rigorous investigation of several performance indicators involved in an AE model. In our proposed model, we use a new data pre-processing methodology that transforms and removes the most affected outliers from the input samples to reduce model bias caused by data imbalance across different data types in the feature set. Our proposed model utilizes the most effective reconstruction error function which plays an essential role for the model to decide whether a network traffic sample is normal or anomalous. These sets of innovative approaches and the optimal model architecture allow our model to be better equipped for feature learning and dimension reduction thus producing better detection accuracy as well as f1-score. We evaluated our proposed model on the NSL-KDD dataset which outperformed other similar methods by achieving the highest accuracy and f1-score at 90.61% and 92.26% respectively in detection.

INDEX TERMS Network security, intrusion detection systems, network-based IDSs, anomaly detection, NSL-KDD, artificial intelligence, machine learning, deep learning, autoencoders, unsupervised learning.

I. INTRODUCTION

The number of devices with the Internet connection are predicted to reach 500 billion by 2030 [1]. The boundless Internet connectivity provides tremendous convenience and opportunities for corporations. However, it also brings serious network security risks and challenges where enormous growths of network intrusions and cybercrimes have been recorded in recent years [2], [3].

To address network security concerns, it is crucial to acquire insight into the pattern of network attacks and to provide resilient solutions to ensure network security. The fast-paced development of data science and Artificial Intelligence (AI) methods has proved to be powerful tools to solve complex challenges and issues. In recent years, there have been many AI-based network anomaly detection methods proposed to demonstrate the feasibility of using data science with AI methods to improve network security concerns.

A deep learning method using Autoencoder (AE) [4] models has recently gained popularity for finding anomalous features contained among the large network traffic samples [5]–[7]. The AE model is suitable for the task of network anomaly detection due to its relatively simple mechanism to train the input and reconstruct the output from it. The training phase of the AE model aims to reduce the

The associate editor coordinating the review of this manuscript and approving it for publication was Ze Ji.

reconstruction loss between the input and output. The rate of reconstruction loss is used to decide whether a network sample is normal or anomalous. There has been a number of AE-based approaches for network anomaly detection by varying a number of performance indicators of the AE model which includes model architecture, adapting different data pre-processing methodologies, the use of different reconstruction loss schemes, etc. However, these existing state-of-the-arts do not offer a holistic approach to examine the impact of the core set of the performance indicators for AE models, report on a solid set of investigation as to what works and whatnot, and proposing the best working AE model for the network anomaly detection task.

We propose a novel 5-layer AE model that is better equipped to accurately identify anomalous network traffic based on the finding from an extensive investigation on the set of core performance indicators of AE model construction.

The contribution of our proposed model is following:

- We confirm that there is a high correlation between the quality of data collection (e.g., input samples) and the detection accuracy. Unlike the data pre-processing methodologies adopted by existing state-of-art AE models, the best accuracy is achieved when data encoding is done before outlier removals and normalization. Our study found that by proceeding with the data encoding as the first step in the data pre-processing, the data balance across different data types is better maintained thus reducing model bias during the model training.
- The percentile rule provides a simple yet effective non-parametric method to identify outliers, which is especially useful for obtaining an adequate reconstruction loss distribution when training an AE model. It also has the flexibility in tuning the model to obtain better performance by changing the percentile in the outlier removal process.
- We validated the impact of different reconstruction loss functions on detection accuracy. Though the difference is not large, the Mean Absolute Error (MAE)-based reconstruction loss function provided the best accuracy for the AE model used in network anomaly detection.
- We studied the impact of performance on different AE-based model architecture. The best performing AE model was the 5-layer model which constitutes of 1 input layer, 2 dense layers, 1 bottleneck layer, and 1 output layer. There was no significant large difference in the performance – less than 5% variation in both accuracy and f1-score range -though the number of hidden layers and the size of neurons were different across different AE architecture. Our experimental result also illustrates that the model architecture has lesser influence on the performance compared to data selection.
- The best performing AE model was achieved in the following performance indicator conditions. (1) When 95 percent of feature wise normal data was used retained after one hot encoding to train autoencoder model, (2) MAE-based reconstruction loss function was used, and (3) 5-layer model architecture with the number of neurons on AE [122-32-5-32-122]. We test our proposed approach on the NSL-KDD dataset [8], which is the most widely used latest public dataset for intrusion detection methods, and obtained the high performance of 90.61% accuracy and f1-score 92.26%, which outperformed other similar methods.

We organize the rest of the paper as follows. We examine the related work in Section II. We provide the details of our proposed AE model along with its architecture and the algorithm in Section III. In Section IV, we provide the details of the NSL-KDD dataset and data pre-processing methodology that work better for AE-based network anomaly detection. In Section V, we describe experimental results including the experiment setup, the description of the performance metrics, and the results. Finally, we provide a conclusion of our work and present future work directions in Section VI.

II. RELATED WORK

Anomaly detection using machine learning techniques has gained popularity in recent years instead of traditional signature-based intrusion detection methods [9], [10]. Due to the automation nature of the machine learning technique, it was now possible to build different machine learning methods without the strong involvement of human domain experts [9] which was often the limitation and expensive.

Depending on the existence of labels in the model training, proposed methods were categorized as either supervised or unsupervised learning algorithms. In the realm of supervised machine learning-based network intrusion detection, the problem becomes a classification task. To identify whether a traffic sample is an attack or not, researchers explored different binary classification algorithms to acquire a highly accurate detection rate. The authors in [8] used the J48 model on the KDD99 dataset to achieve the accuracy of 93.82% and Naïve Bayes Tree (NBTree) on the NSL-KDD dataset of 82.02% accuracy. A number of methods using Decision Tree (DT), Naïve Bayes Network (NB), and Support Vector Machine (SVM) were introduced for network anomaly detection by [11]. The authors in [12] employed Fuzzy logic in anomaly detection and obtained an accuracy of 84.54% in the experiment. The authors [13] proposed an Artificial Neural Network (ANN) model and reported 81.2% accuracy on the NSL-KDD dataset. Hybrid models by combining different state-of-the-arts algorithms to deliver an improvement on the detecting performance were also proposed. For example, Kevric *et al.* [14] illustrated that combining two tree algorithms gain better performance than individual tree classification while they reported the best combination is the random tree and NB tree with the accuracy of the 89.24% on the KDD dataset. Autoencoder (AE) which commonly used for feature extraction has been widely used in the first stage of hybrid models. A benefit of using AE is that it generates a condensed representation of the original input by removing noise from it [9], [15], [16]. Azar *et al.* [17] used AE for feature learning then used supervised machine learning algorithms such as SVM, KNN for classification to achieve 83.3% accuracy. Similarly, Al-Qatf *et al.* [7] combined AE and SVM together and obtained the 84.96% accuracy rate on the KDD dataset for binary classification. Their proposed approach also used an AE to reduce dimension and learns the feature representation. Javaid *et al.* [18] proposed sparse-autoencoder for feature learning and soft-max regression-based neural architecture for classification and they achieved 88.39% accuracy in intrusion detection. Though the supervised learning approaches (include hybrid ones) had gained high performance in numerical results, their success highly relied on correct labels and balanced data in the training process, which means they could only efficiently classify unseen samples by training with a large amount of similar data with corresponding labels [10]. However, in the network intrusion detection field, very little intrusion data is publicly available due to complex reasons, e.g. privacy issues and data confidentiality [19]. To address the limitation, unsupervised learning methods (e.g., Autoencoder (AE)) using anomaly detection approaches have been introduced only to use benign samples in the training phase. Ieracitano *et al.* [20] analyzed the NSL-KDD dataset with a statistical approach and tested it with a simple 3-layer AE architecture. They obtain the value of 84.21% accuracy in binary classification. The authors in [21] automated threshold learning for anomaly detection in an autoencoder-based model and achieved a high of 88.98% accuracy.

The majority of these existing works [7], [12], [13], [20] use encoding mechanism for categorical (nominal) features in the dataset after different data pre-processing procedures when processing the features of the NSL-KDD dataset. We argue that their methodologies introduce data imbalance issues because they remove categorical values too early in the data pre-processing stage which significantly affects the performance of proposed models. The studies in [12], [13] analyze the features of the input samples using different clustering mechanisms applied for detecting the most optimal number of outliers and to reduce the dimension of features. We argue that these methods are not applicable and generalizable to apply to other datasets in similar models. The study done by [20] only analyzes the outliers in the numerical features by leaving the symbolic features untouched. We argue that this also creates a bias because most likely symbolic features also have outliers and these need to be handled properly.

III. AUTOENCODER-BASED NETWORK ANOMALY DETECTION

A. GENERIC MODEL

An autoencoder (AE) is an unsupervised feed-forward neural network used for the reconstruction of its input. An AE composes of an input layer, an output layer, and one or more hidden layers. It has a symmetrical pattern – the output layer has the same number of neurons as the input layer while any hidden layer generally has fewer neurons than the input and output layer.

The bottleneck layer, also referred to as a latent space, is one of the hidden layers which has the smallest number of neurons. The latent space contains the compressed representation of the input. The mechanism of autoencoder attempts to reconstruct the input at the output, to receive a similar input and output, i.e. $\hat{x} = x$. An example of a generic autoencoder is shown in Fig.1.

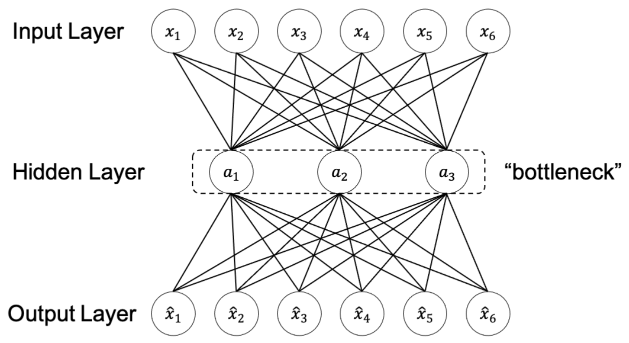

Input Layer
x_1 x_2 x_3 x_4 x_5 x_6
Hidden Layer
a_1 a_2 a_3
"bottleneck"
Output Layer
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$

FIGURE 1. A generic autoencoder model.

A generic autoencoder architecture consists of two operations, encoding and decoding respectively.

In the encoding operation, any input sample x is an m dimensional vector $[x_1, x_2, x_3, \ldots, x_m]$ and is mapped to the hidden layer representation (y), as shown in equation (1).

$$y = f_1(wx + b) \tag{1}$$

where f_1 ia an activation function for the encoder. w represents the weight matrix, and b is a bias vector.

In the decoding operation, the hidden representation of (y) is mapped back into a reconstruction $\hat{x}$, as shown in equation (2).

$$\hat{x} = f_2(w'y + b') \tag{2}$$

where f_2 is an activation function for the decoder. w' and b' represents the weights and bias for the output layer.

During these two operations, the neural network's parameters $\theta = (w, w', b, b')$ are continuously optimized by minimizing the reconstruction error. To minimize reconstruct error on x with non-linear functions, the loss reconstruction (L) is calculated from equation (3).

$$L(x, \hat{x}) = \frac{1}{m}\sum_{i=1}^{m}(x_i - \hat{x}_i)^2 \tag{3}$$

B. OUR MODEL

The AE model in the network anomaly detection tasks use the reconstruction error to find whether a network traffic sample is anomalous or not. In intuition, a network sample showing high reconstruction error during the testing phase should be considered anomalies when an AE trained on a normal network traffic dataset presents low reconstruction error. Therefore, our proposed model is built on this concept - details are in Algorithm 1.

Algorithm 1: AE-based Network Anomaly Detection

```
Input: Training dataset $S = \{X_1, X_2, X_3, \dots, X_n\}$
Testing dataset $N = \{X'_1, X'_2, X'_3, \dots, X'_n\}$
  /* X and X′ are both m dimensional vectors */
    Encoder $E_\phi$; Decoder $D_\theta$
Output: AnomalySet, NormalSet
begin
    /* Step 1: Training Phase */
    $\phi, \theta \leftarrow$ Initialize parameters
    /* Training in mini-batch */
    for number of training iterations do
        sample mini-batch of $k$ samples
          $\{X_1, X_2, X_3, \dots, X_k\}$ from $S$
        /* Calculate sum of mini-batch loss */
        $V(E, D) = \frac{1}{m} \sum_{i=1}^{k} (X_i - D_\theta(E_\phi(Xi)))^2$
        $\phi, \theta \leftarrow$ Update parameters using Stochastic
          Gradient Descent of $V(E, D)$
    end
    /* obtain Threshold from Training dataset */
    for each $X \in S$ do
        $\hat{X} = D_\theta(E_\phi(X))$
        /* reconstruction loss metric: MAE */
        $L(X, \hat{X}) = |X - \hat{X}|$
    end
    Threshold $\alpha = \max(L)$    /* Threshold */
    /* Step 2: Testing Phase */
    for each $X' \in N$ do
        $L(X') = |X' - D_\theta(E_\phi(X'))|$
        if $L(X') > \alpha$ then
            $X'$ is an anomaly
            insert $X'$ to AnomalySet
        else
            $X'$ is NOT an anomaly
            insert $X'$ to NormalSet
        end
    end
end
```

In the training phase, the original features of the network traffic are extracted and reduced by the encoding operation then represented in the latent space. The latent space is then used to reconstruct the output. The difference between the output traffic sample and the original traffic sample is compared and a reconstruction error is computed. Once all traffic samples are processed by the model, the max value of all reconstruction errors is marked as the threshold to identify anomalies.

During the testing phase, network traffic samples are inputted to the trained AE model and again a reconstruction error is calculated – it is called an anomaly score now. The anomaly score is compared with the threshold value obtained during the training phase. If the anomaly score is larger than the threshold, this traffic sample is now considered anomalous.

In this study, we propose a 5-layer AE architecture. The AE encodes the 122-dimensional features representation (x) into a 32-dimensional vector (m) which is further reduced as a 5-dimensional vector (a) and then decodes it back to the same input features space. The proposed AE [122-32-5-32-122] is trained in an unsupervised manner, using mini-batch stochastic gradient descent. All the hidden layers are dense layers (i.e., fully connected layer that connects all neurons from the previous layer) using rectified linear units (ReLU) as activation function instead of sigmoid function in the compression and reconstruction operations for faster computation. The reconstruction error between x and $\hat{x}$ is quantified using MAE value. Fig. 2 demonstrates the architecture of our proposed model.

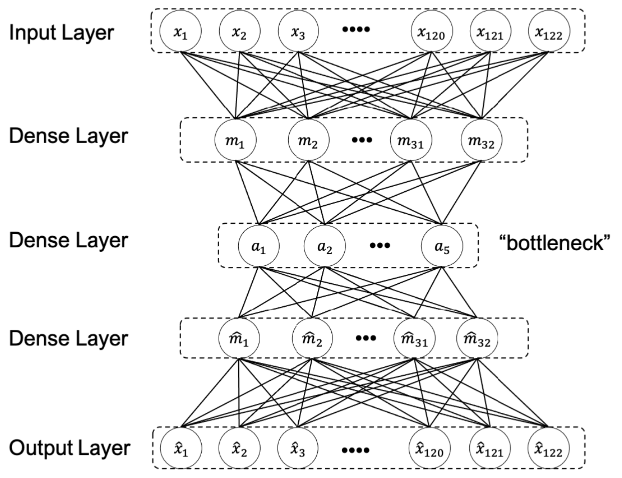

Input Layer
x_1 x_2 x_3 •••• x_{120} x_{121} x_{122}
Dense Layer
m_1 m_2 ••• m_{31} m_{32}
Dense Layer
a_1 a_2 ••• a_5
"bottleneck"
Dense Layer
$\hat{m}_1$ $\hat{m}_2$ ••• $\hat{m}_{31}$ $\hat{m}_{32}$
Output Layer
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ •••• $\hat{x}_{120}$ $\hat{x}_{121}$ $\hat{x}_{122}$

FIGURE 2. **Our proposed 5-layer AE model.**

IV. DATA AND METHODOLOGIES

In this section, we describe the details of the data we use for our study (i.e., NSL-KDD), the methodology we employed for data processing, and the workflow of our proposed model.

The NSL-KDD dataset has two datasets, KDDTrain+ and KDDTest+, respectively. Though both datasets contains both normal and abnormal network traffic samples, we only use the normal network traffic samples from the KDDTrain+ for training.

As seen in Fig. 3, we first use only the KDDTrain+ dataset after applying a number of data pre-processing techniques such as one-hot-encoding to transform the categorical features into numeric data, disposal of outliers, and normalizes the dataset by scaling them to fit in the range of [0, 1]. After pre-processing the KDDTrain+ dataset, we fit the dataset into our proposed AE model which computes the threshold (i.e., reconstruction error rate associated with normal traffic pattern). At the testing phase, the KDDTest+ dataset is used on the trained AE to calculate an anomaly score (i.e., the same meaning as the threshold that calculator reconstruction error). The underlying assumption is that the reconstruction error rate calculated for the normal traffic must differ from the

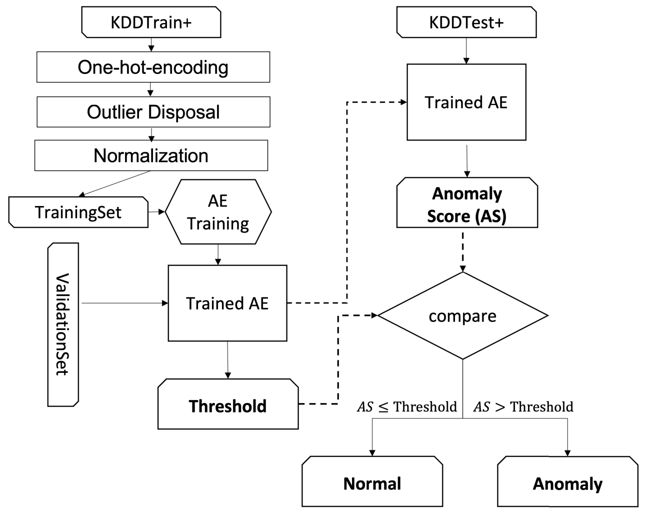


FIGURE 3. Workflow of our proposed model.

anomalous traffic as the feature values must differ between them. Under the assumption, the anomaly score is then compared with the threshold to determine whether a test sample is a normal network traffic (i.e., the anomaly score stay under the threshold) or an anomalous one (i.e., the anomaly score goes beyond the threshold).

Notice that the encoding process is put ahead of the outlier disposal in the data pre-processing stage. By doing this, now the categorical features and the numerical features are treated equally and the outliers contained in the categorical features are now handled appropriately which reduces the biases in data balance.

A. NSL-KDD DATABASET

NSL-KDD is a dataset suggested to solve many inherent problems [8] associated with earlier versions (e.g., KDDCup99) used for network intrusion detection. Though the dataset may not be a perfect representative of existing real networks, because of the lack of public datasets for network-based IDSs, it is often regarded as the most widely used latest network intrusion datasets that can be applied as an effective benchmark to compare different intrusion detection methods along with UNSW-NB15 and CICIDS-2017.

We use two subsets of the NSL-KDD datasets, KDDTrain+ and KDDTest+ respectively, for AE model training and evaluation. Though both KDDTrain+ and KDDTest+ contain multiple class labels, we re-classify them into two categories, whether the traffic sample contained in these datasets are normal and abnormal to focus on the impacts of the major performance indicator.

As illustrated in Table 1, the KDDTrain+ dataset contains the total of 125,973 records in which 67,343 of them are labelled as "normal" while 58,630 are labelled as "abnormal". Similarly, the KDDTest+ contains a total of 22,544 records of which 9,711 of them are labeled as

TABLE 1. Records of two NSL-KDD datasets: KDDTrain+ and KDDTest+.

NSL-KDD	Total	Normal	Others
KDDTrain+	125,973	67,343	58,630
KDDTest+	22,544	9,711	12,833

"normal" while the rest of 12,833 are labeled as "abnormal" traffic samples.

Fig. 4 illustrates the visualization of the NSL-KDD dataset using Principal Component Analysis (PCA). Fig. 4 (a) shows the distribution of the normal and abnormal samples in the KDDTrain+ dataset. It is visible that there is a good balance in the number of traffic samples between normal and abnormal. The features of each class of the traffic sample appear to be similar as a clear distinct cluster is seen around the normal dataset while there are 3 different clusters formed around abnormal datasets. Fig. 4 (b) shows the distribution of the normal and abnormal samples in the KDDTest+ dataset. There are more abnormal datasets compared to the normal dataset. The features in the abnormal traffic sample appear to be distinct from each other as there are no visible clusters formed around.

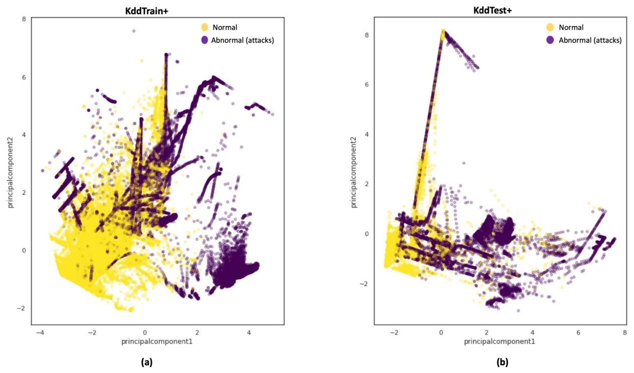


FIGURE 4. The visualisation of PCA for NSL-KDD dataset.

Each traffic sample in the NSL-KDD dataset contains a total of 41 features, including 38 numeric (e.g., "int64" or "float64") and 3 categorical values (e.g., "object"). Table 2 shows the name and data type of all 41 features.

B. DATA PRE-PROCESSING

We go through three different data pre-processing procedures to organize and transform the NSL-KDD datasets before feeding into the AE model. These include: one-hot-encoding, outlier disposal, and min-max normalization.

1) ONE-HOT-ENCODING

To increase the efficiency of model training, AE models demand non-numerical features (e.g., categorical values) converted into numerical values. We use the one-hot-encoding technique to convert categorical features into n-dimensional vectors of binary code, where the "n" is determined by the total number of attributes in the categorical feature. Take the

TABLE 2. NSL-KDD dataset features: 38 numeric and 3 categorical.

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			

feature "protocol_type" in NSL-KDD dataset for example where there are three distinct attributes "tcp", "udp" and "icmp" each of which are encoded into three 3-dimensional binary vectors: [1,0,0], [0,1,0], [0,0,1] respectively. In other words, the single feature 'protocol_type' is encoded into three features by one-hot-encoding. In the NSL-KDD dataset, there are three categorical features (namely "protocol_type", "service", and "flag") each of which has 3, 70, and 11 distinct attributes respectively. These are converted into a total of 84 features. Combined with the 38 numerical features, now we have a total of 122 features produced after the one-hot-encoding is applied.

2) OUTLIER ANALYSIS

An outlier is a data point that differs significantly from other data points in a dataset [22]. The source of outlier varies. In our study, we specify an outlier if a feature in a dataset contains an extreme value that deviates from what we consider from the "normal" range. It is important to remove such outliers because they tend to generate bias on the correct calculation of weights. This makes AE models less sensitive to anomalies, thus consequently, decreases the accuracy of anomaly detection. To address this issue, we remove outliers before model training.

Towards outlier disposal, the first and foremost step is to identify outliers. Several outlier detection methods in statistics have been introduced in the literature. Tukey's fences [23] is one of the common methods used for outliers detection as it calculates the outlier fence with the use of interquartile range (IQR). The formula is depicted as follow:

$$[Q_1 - k(Q_3 - Q_1), Q_3 + k(Q_3 - Q_1)] \tag{4}$$

where Q_1, Q_3, and k represent the lower quartile, upper quartile, and the coefficient respectively. If the coefficient $k = 1.5$ and the test data is out of the IQR range, the test data will be regarded as an "outlier", and $k = 3$ means the data is "far out". However, this method alone is not practicable for the KDDTrain+ dataset because the distribution of the dataset is extremely imbalanced. In fact, 21 out of 38 numerical features of the KDDTrain+ dataset have both Q_1 and Q_3 equal to the minimum value of zero. Hence, a massive number of mislabelled outliers may produce.

Another popular outlier analysis method is Z-scores [24], [25]. Z-score is calculated with the following formula:

$$Z_i = \frac{(X_i - \overline{X})}{\sigma} \tag{5}$$

where $\overline{X}$ and σ are the mean and standard deviation of the distribution of the feature X, and X_i is the attribute of i^{th} sample in that feature. Z-score assumes that the feature is independent with other features and the distribution of the feature is subordinate to normal distribution. Three-sigma rule [26], also called the 68-95-99.7 rule, are applied for outlier identification with Z-score in general. The rule expresses that about 68% of the instances lie in one sigma (or standard deviation) of the mean value, and about 95% instances in two sigmas while about 99.7% in three sigmas.

For our study, we adopted the outlier fence concept and choose the variation of the two-sigma (95%) effect for outlier detection. The proposed outlier detection method is called the 95th percentile rule - any sample has an attribute greater than the 95th percentile of all instances in that feature is regarded as outliers. All identified outliers are removed from the dataset afterward. The pseudocode 2 depicts the process of the proposed outlier disposal.

Algorithm 2: Outlier Disposal

```
S: samples of dataset $\{s_1, s_2, \ldots, s_m\}$
F: features in samples $\{f_1, f_2, \ldots, f_n\}$
Calculate upper outlier fence of features
  $OF = \{of_1, of_2, \ldots, of_n\}$
for each $s \in S$ do
    for each $f \in F$ do
        if $s_j.f_i > of_i$ then
            break
        end
    end
    delete $s_j$
end
```

Our hybrid approach of outlier removal has three distinct advantages compared to other similar statistical methods. Firstly, our hybrid outlier removal approach makes no assumption about the distribution of samples so the method can be applied to any dataset. Secondly, the 95th percentile is the upper outlier fence, and no lower outlier fence is set in the experiment due to the analysis of the distribution of the KDDTrain+ dataset. As mentioned earlier, 75% samples with numerical values have the minimum value 0 so the lower outlier fence is equal to the minimum value 0. In other words, no lower outlier fence is necessary. The last advantage is that

TABLE 3. Confusion matrix.

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

we identify outliers after encoding the categorical features, which means the outlier detection rules are applied to the hot encoded features as well.

Note that the outliers only in the training dataset are removed because only ''normal'' samples in the KDDTrain+ are used for the model training. Its sample size has changed from 67,343 to 39,252 after our hybrid approach was applied.

3) DATA NORMALIZATION

Normalization eliminates the impacts of different scales across features thus reduces the execution time for model training. The min-max normalization is applied after outliers are moved. This method maps the original range of each feature into a new range with the formula:

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{6}$$

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

where min, max $= (0, 1)$ in default are used in this experiment to normalize all numerical features [27].

V. EXPERIMENTAL RESULTS

In this section, we provide the details of the performance metrics used in our experiment and the analysis of results.

A. PERFORMANCE METRICS

To evaluate the performance of our proposed model, we use classification accuracy, precision, recall, and F1 score. We follow the convention which labels the normal samples as class 0 while the anomalous samples as class 1. Table reftable:Matrix illustrates the confusion matrix, where True Positive (TP) is the number of correctly labeled cases for class 1 (in our case anomalous traffic samples), True Negative (TN) is the correctly labeled class 0 cases (in our case normal traffic samples), False Positive (FP) is class 0 cases that are incorrectly labeled as class 1 and False Negative (FN) class 1 samples but miss-classified as class 0.

True Positive Rate (TPR) is also called Recall or sensitivity, which indicates the proportion of data points correctly classified as anomalous data points, as shown in Equation 8.

$$TPR/Recall = \frac{TP}{TP + FN} \tag{8}$$

Precision (Pr) denotes the proportion of TP data points, which is also known as a positive predictive value, as shown in Equation 9.

$$Pr = \frac{TP}{TP + FP} \tag{9}$$

Accuracy (Acc) measures the proportion of correct prediction and indicates the proportion of the number of correctly classified data points to total data points for a given dataset in Equation 10.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

F1-score (F1) denotes the measure of the harmonic mean of recall (or TPR) and precision on Equation 11.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{11}$$

B. RESULTS

We have made a number of different observations to understand the performance implications both during the training and testing phases.

1) DATA REPRESENTATION AT THE LATENT SPACE

Our model has been trained with 80% of the training dataset (presented as TrainingSet in Fig. 3) while validated with the left 20% (ValidationSet in Fig. 3) for 50 epochs. The training dataset is shuffled at the beginning of each epoch to avoid overfitting.

The visualization of the distribution between the normal and abnormal samples across KDDTrain+ and KDDTest+ in the latent space, in which the data lies in the bottleneck layer, is shown in Fig. 5. The latent space contains a compressed representation of the traffic samples which is the only information the decoder uses to try to reconstruct the input. For the model to perform well, it has to learn to extract the most relevant features in the bottleneck. Fig. 5 (a) shows two distinct clusters, one clearly belongs to the normal samples and the other abnormal samples clustered yet widely scattered around while Fig. 5 (b) shows the normal samples scattered wildly across wider space with no visible cluster formed around.

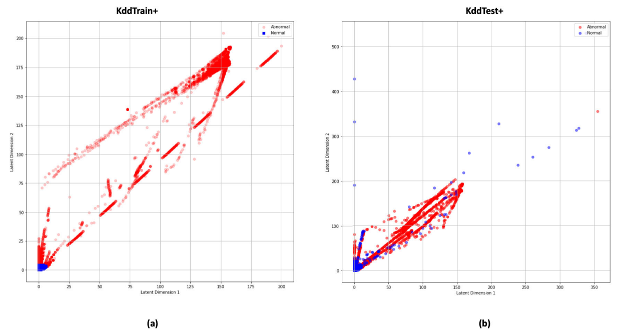


FIGURE 5. The visualization of latent space representation of KDDTrain+ and KDDTest+.

2) IMPACT OF RECONSTRUCTION LOSS FUNCTIONS

The aim of this experiment was to understand the sensitivity of different reconstruction loss functions to the detection accuracy. The three reconstruction loss functions

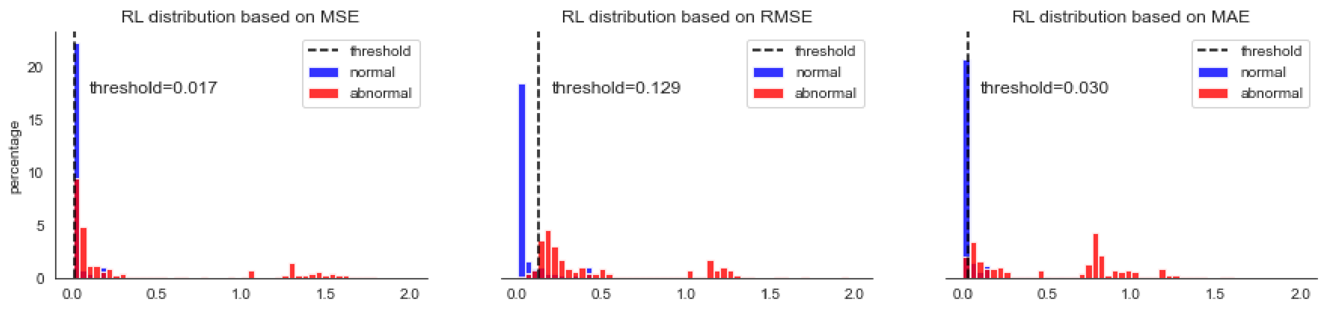


FIGURE 6. Threshold and the distribution of reconstruction loss from different matrixs of KDDTest+, reconstruction loss range within [0,2].

were studied: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Squared Error (MSE), respectively. The definitions of these functions are described in the following Equations.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}(x_i - \hat{x}_i)^2}{n}} \tag{12}$$

$$MAE = \frac{\sum_{i=1}^{n}\left|x_i - \hat{x}_i\right|}{n} \tag{13}$$

$$MSE = \frac{\Sigma_{i=1}^{n}(x_i - \hat{x}_i)^2}{n} \tag{14}$$

where n indicates the total number of traffic samples, x_i is the representation of the original input sample while $\hat{x}_i$ is the output represented at the latent space.

Table 4 illustrates that there are variations of the threshold values computed depending on the reconstruction loss function used. Though there are differences in the threshold values, both RMSE and MSE provide identical values in the four different performance metrics. Though the differences are not visible, MAE came out as the best reconstruction loss function works best for the AE models used in the network anomaly detection compared to the other two functions.

TABLE 4. Performance of different loss mechanisms.

Metric	Threshold	Accuracy	Precision	Recall	F1 score
RMSE	0.129	90.47%	86.79%	98.21%	92.14%
MAE	**0.030**	**90.61%**	**86.83%**	**98.43%**	**92.26%**
MSE	0.017	90.47%	86.79%	98.21%	92.14%

To examine the effect of reconstruction error function more closely, Fig. 6 visualizes the relationship between thresholds and the range of reconstruction error computed across the network traffic samples in the KDDTest+ dataset labeled between normal and abnormal. We can clearly see that the reconstruction error computed for the network traffic samples which are labeled as "normal" strongly correlates around the threshold – the majority of their reconstruction error is below and on par with the threshold. On the contrary, we see that the reconstruction errors for the network traffic samples with the label "abnormal" tend to spread widely where the majority of the values are bigger than the threshold.

3) IMPACT OF OUTLIERS

Next, we studied the relationship between the percentile of outliers in the input samples on the KDDTest+ and the accuracy. As seen in Table 5, the detection accuracy improved as the number of outliers were removed until the 95th percentile rule was applied which peaked the accuracy at 90.61% when the 5% of outliers were removed from the input samples.

TABLE 5. Performance with different percentiles on KDDTest+.

Percentile	Threshold	Accuracy	Precision	Recall	F1 score
99	0.052	83.05%	92.27%	76.64%	83.73%
98	0.054	86.04%	92.61%	82.02%	87.00%
97	0.064	85.25%	91.62%	81.55%	86.29%
96	0.036	88.82%	88.03%	93.00%	90.45%
95	**0.030**	**90.61%**	**86.83%**	**98.43%**	**92.26%**
94	0.042	90.53%	89.24%	94.80%	91.94%
93	0.037	90.36%	86.13%	98.99%	92.12%
92	0.034	87.15%	83.12%	97.17%	89.60%
91	0.052	89.07%	86.27%	96.10%	90.92%
90	0.033	86.88%	81.27%	99.99%	89.67%

4) IMPACT OF MODEL ARCHITECTURE

In this experiment, we examined the impact of different architecture and the accuracy. We experiment on 3-layer (i.e., 1 input layer, 1 bottleneck layer, 1 output layer), 5-layer (i.e., 1 input layer, 2 dense layers, 1 bottleneck layer, 1 output layer), and 7-layer models (i.e., 1 input layer, 4 dense layers, 1 bottleneck layer, 1 output layer) that were used by the state-of-the-art in AE models. Across different hidden layers, we also variate the number of neurons at different layers.

As illustrated in Table 6, the results show that there is a little difference in performance across different model architecture by the accuracy rate close within the 5% from the least performing model of the 3-layer model with 1 bottleneck layer with the 5 neurons at [5] to the best performing model of the

TABLE 6. Performance of AE with different model architecture.

architecture	Accuracy	Precision	Recall	F1-score
[5]	86.52%	89.35%	86.65%	87.98%
[10]	88.75%	88.36%	92.40%	90.34%
[32,5,32]	**90.61%**	**86.83%**	**98.43%**	**92.26%**
[32,10,32]	87.44%	87.84%	90.46%	89.13%
[64,5,64]	89.32%	88.54%	93.31%	90.86%
[32,16,5,16,32]	88.30%	87.09%	93.28%	90.08%
[64,32,5,32,64]	89.03%	86.79%	95.23%	90.82%

5-layer with 2 hidden and 1 bottleneck layer with the size of neurons at [32-5-32]. This result is very similar to the finding from another study [20] which also illustrated the accuracy range was within the 5% from 79.56% to 84.24% when more than 20 different architectures were tested. This confirms that the sensitivity of the accuracy against the model architecture is lower than the sensitivity of the accuracy against data selection (e.g., outliers).

In addition to the performance results, we explored which features were considered important by our model for deciding whether a traffic flow was normal or abnormal. Note that due to the complexity and "black-box" nature involved in deep learning models, it is not possible to map back from the features selected at the latent space to the original features. To address this, we calculated the absolute error of each feature between the original sample x and its reconstruction $\hat{x}$ in KDDTrain+ once the model was trained. This was followed by computing the Pearson correlation coefficient [28] between the feature-level error and the corresponding label. By doing this, we could acquire the importance of each training feature that was considered by our model. Fig. 7 presents the top 5 features most strongly associated to contribute the detection which includes: 2 different types of "flag" (i.e., flag_RSTR, flag_RSTOS0), "service_whois", "dst_host_serror_rate", and "srv_serror_rate".

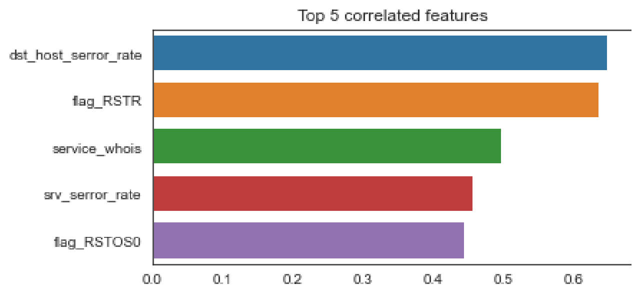


FIGURE 7. The top 5 correlated features within 122 encoded features: "dst_host_serror_rate", "flag_RSTP", "service_whois", 'srv_serror_rate' and 'flag_RSTOS0'.

5) COMPARISON WITH OTHER SIMILAR METHODS

We compared the performance of our proposed model with other similar models. We compared the performance by using the four metrics, namely accuracy, precision, recall, and F1-score. The table 7 illustrates that our proposed AE model can obtain an accuracy of higher than 90% and the highest f1-score 92.26%.

TABLE 7. Performance comparison with other approaches on KDDTest+.

Method	Accuracy	Precision	Recall	F1-score
AE by Azar et al. [17]	83.34%	-	-	-
Sparse-AE + SVM [7]	84.96%	96.23%	76.57%	85.28%
AE by Ieracitano et al. [20]	84.21%	87%	80.37%	81.98%
AE by Javaid et al. [18]	88.39%	85.44%	95.95%	90.4%
AE by Sadaf et al. [21]	88.98%	87.92%	93.48%	90.61%
Our proposed method	**90.61%**	**86.83%**	**98.43%**	**92.26%**

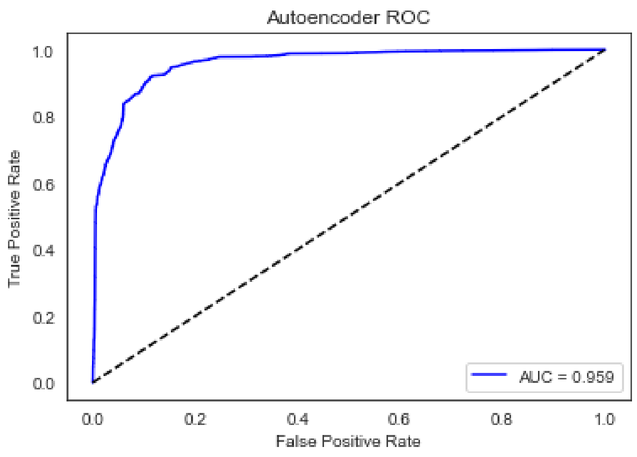


FIGURE 8. ROC curves of proposed autoencoder.

Fig. 8 demonstrates the ROC curve (receiver operating characteristic curve) of our proposed model. Our model confirms its good performance by producing the AUC (area under the ROC curve) score ($AUC_{AE} = 0.959$) with a very high true positive rate along with a low false-positive rate. As pointed out by Sommer and Paxson [19], though our model exhibits a low false-positive rate compare to other similar methods, such a low rate can still bring serious consequences in some high-security reliant applications. Techniques such as explainable AI (xAI) [29] can be further applied to diagnose, debug, and improve the model to remove as many false-positive as possible.

VI. CONCLUSION

We propose a novel 5-layer AE-based model better suited for detecting anomalous network traffic. The main components and architecture of our proposed model are obtained from the result of an extensive and rigorous study by examining the impact of the major performance indicators of an AE model and the detection accuracy. Our experimental results show that our proposed 5-layer architecture model achieves the highest accuracy with the proposed two-sigma (95^{th} percentile) outlier disposal approach and MAE as reconstruction loss metric.

Our model uses an innovative data pre-processing methodology that effectively transforms the input datasets to contain more balanced data samples in terms of data type and data size as well as removes outliers that would most likely affect the detection bias. The effectiveness of the proposed data pre-processing methodology was obtained by analyzing the impact of the percentile rule in the outlier disposal stage. Our model utilizes MAE as the basis of the reconstruction loss function which provides the best accuracy for the AE model used in the network anomaly detection. Our 5-layer architecture with the optimized number of neurons used at each hidden and latent space layer provides the best performance compared to other model architecture. We evaluated our proposed model on the widely used NSL-KDD dataset. The test results demonstrate that our approach generates the best performance at the 90.61% accuracy, 86.83% precision, 98.43% recall, and 92.26% F1-score compared to other similar models.

Our experimental results also confirm that among the performance indicators of an AE model, which includes data pre-processing, reconstruction loss metric, and model architecture, data pre-processing has the largest effect on the performance. Though currently trained on NSL-KDD dataset, our proposed model is equipped and tested to recognize any abnormal network traffic pattern deviating from normal traffic patterns, very efficiently. Though the characteristics of intrusion samples may differ in other intrusion datasets, we believe our model can still work very well in detecting any abnormal patterns. However, further studies are required to test how effectively our proposed model can work in real-world large-scale operational network environments by incorporating deeper semantic insights into real systems' capabilities and limitations.

We have plans in place to apply different types of intrusion attack samples (e.g., Android-based malware samples [30] or ransomware [31], [32]) and other dataset samples from other applications (e.g., indoor air quality (IAQ) [24], [25], [33], medical annotations) to test the generalizability and practicability of our model. We also plan to extend our current work to multi-class classification.

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. (IJCNN)*, 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 PM_{10} 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.

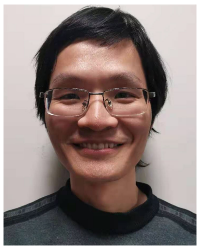
[29] M. T. Ribeiro, S. Singh, and C. Guestrin, '''Why should I trust you?' Explaining the predictions of any classifier,'' in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1135–1144.

[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.



WEN XU received the master's degree in information science from Massey University, Auckland, New Zealand. He is currently a Junior Research Officer with the School of Natural and Computational Sciences, Massey University. His current research interests include deep learning and AI-based network intrusion detection.

JULIAN JANG-JACCARD received the M.Sc. and Ph.D. degrees from The University of Sydney, Australia. She is currently an Associate Professor and the Head of the Cybersecurity Laboratory, Massey University, New Zealand. She has published more than 70 papers in the leading conferences and journal venues, including IEEE and ACM. Her research interests include cybersecurity, intrusion detection, anomaly detection, artificial intelligence, data anonymization, and privacy-preservation techniques. She was a recipient of many multi-million dollar research awards both from Australian and NZ governments/industries while collaborating with the top international ICT companies and universities around the world.

AMARDEEP SINGH received the Ph.D. degree from Massey University, New Zealand, in 2021. He is currently a Research Officer at Massey University, where he is working on providing AI solutions for network intrusion/malware detection and classification. He is also involved in research on explainable AI to make AI solutions trustworthy and secure for cybersecurity.

YUANYUAN WEI received the master's degree in information technology from Massey University, Auckland, New Zealand, where she is currently pursuing the Ph.D. degree with the School of Natural and Computational Sciences. Her research interests include AI-powered anomaly detection, network intrusion detection, machine learning, and deep learning.

FARIZA SABRINA (Member, IEEE) received the M.E. degree (by Research) in electrical and information engineering from The University of Sydney, Australia, and the Ph.D. degree in computer science and engineering from the University of New South Wales. She is currently a Senior Lecturer at ICT, School of Engineering and Technology, Central Queensland University, Australia. She has authored/coauthored and published many papers in top-ranking conferences and journals. Her research interests include cybersecurity, the Internet of Things, network security, blockchain, artificial intelligence, machine learning, and deep learning.