ANOMALY & ATTACK DETECTION IN IOT SENSOR DATA USING GATED RECURRENT UNIT AND PARTICLE SWARM OPTIMIZATION TECHNIQUES

VIPIN[1], Professor (Dr.) Mukesh Singla[2]

[1]Research Scholar, [2]Head and Dean

[1,2]Department of Computer Science & Engineering

[1,2]B.M.U. Rohtak, Haryana

Abstract: Internet of Things (IoT)-connected technologies are becoming more and more important to a number of public and commercial businesses. The integrity of data and the availability of services are often the targets of security threats that target the networks and devices that make up the IoT. Due to the various Internet of Things devices and disruptions that are seen inside the IoT system, it is very difficult to identify anomalous behaviour and hacked nodes. This is because of the diversity of data that is obtained from these devices. When compared to contemporary wireless networks, this problem is more difficult. Consequently, reliable anomaly detection systems are critically needed to filter out potentially dangerous data in decision support systems driven by the Internet of Things. This study proposes a solution for anomaly detection within the Internet of Things (IoT) by employing Particle Swarm Optimization (PSO) and Gated Recurrent Unit (GRU), it can train and extract relevant features.

Keywords: Internet of Things, Anomaly detection, IoT security, Machine learning, Deep learning, Cyber security**,** Gated Recurrent Unit etc.

1. Introduction

The IOT is quickly becoming a core component of many different types of technical systems. Innovations such as smart city applications, autonomous autos, and wearable health devices emerged as a result. Sensors embedded in IoT devices transmit data to the cloud, where cyber-physical systems analyze and make decisions based on the collected data. More than 26 billion IoT devices are now online, according to most studies, and predictions show that number will rise to 75 billion by 2025. When it comes to improving productivity and ensuring worker safety, organisations rely heavily on the IoT. Businesses use IoT-based solutions to handle massive volumes of

data produced by IoT-enabled devices. Because of this, they are able to make great strides in improving workplace safety and efficiency [3,4].

With the physical layer in charge of data collecting from IoT sensors, the network layer facilitating data transmission and processing via edge and cloud communication, the processing layer performing operations and evaluations using cloud computation, and the application layer interacting with the end user's device make up the Internet of Things architecture. Despite their interconnectedness, these layers are separate from one another. There are security holes in each of these levels [5-8]. Consequently, in recent years, a lot of focus has been on researching ways to secure IoT devices [9]. The data created, gathered, and processed by IoT devices is very sensitive, which poses a significant risk of security breaches that hackers might exploit. This highlights the critical need of developing trustworthy anomaly detection algorithms that use real-time data gathered from IoT devices[10].

One of the primary functions of learning machines is to develop algorithms that can use benign and dangerous data to detect anomalies and malicious activity in IoT networks. The identification of potentially hazardous content in research publications has been the subject of several machine learning approaches ([11, 12]). But, these methods frequently assume that the training data is homogeneous, meaning that it comes from the same places and has the same properties, like pixel-based image data. Data formats such as text, pictures, time series, streaming data, and graph topologies are more consistent than real-world IoT data. Because of this, heterogeneity is the norm, and traditional machine learning methods fail miserably when faced with IoT data. Also, they aren't advanced enough to handle the complicated data that the Internet of Things produces just yet. Deep learning has the potential to outperform conventional machine learning methods when presented with ever bigger and more diverse datasets [13]. This is indeed doable since deep learning has the potential to use several data layers. A growing number of industries are beginning to recognize the possibilities presented by deep learning's improved anomaly detection methods.

One of the most important parts of processing data from sensors connected to the Internet of Things (IoT) is looking for anomalies [14–16]. There has been a dramatic uptick in the amount of data produced by IoT devices as their number continues to skyrocket. Preprocessing and analysis using AI and data mining techniques is crucial

for extracting relevant information from this data. In this study, we use a number of techniques, including pre-processing, feature selection/extraction, clustering, and GRU modelling, to get data from IoT sensors ready to build anomaly detection models.

2. Literature review

This section delves into how deep learning and machine learning may be applied to the problem of anomaly detection. Several academic studies have proposed using machine learning for outlier identification [17,18]. Anomaly detection may be accomplished by machine learning researchers using one of three separate approaches: supervised learning, semi-supervised learning, or unsupervised learning. Unlike supervised ML models that learn from labelled datasets, unsupervised ML algorithms for anomaly detection derive insights from unlabelled datasets [19, 20, 21, 22]. Finally, the semi-supervised approach is a kind of machine learning that utilizes both unlabelled and labeled data [23,24]. This is a survey of current methodologies for anomaly detection utilizing machine learning.

Lopez-Martin et al. [26] propose a solution for IoT anomaly identification based on a conditional variational autoencoder. Diro and Chilamkurti [27] presented a method for identifying anomalies in the Internet of Things. When it comes to IoT model risks, this method uses fog computing to solve them. Proving that deep learning algorithms are more effective at detecting cyber risks than conventional machine learning models is the main objective of their study.

A combined intrusion detection system using naive decision trees, Bayes and artificial neural networks was studied by Moustafa et al. [28]. With the use of a classifier, this platform can identify botnet assaults on DNS, MQTT, HTTP, and other Internet of Things protocols. Researchers did this by evaluating possible methods and then used current data to create new statistical flow characteristics [28]. In order to detect IoT irregularities, Aversano et al. [29] had an idea for a method that used deep learning. Reducing features with the use of autoencoders improves detection accuracy. In his technique for analyzing Internet of Things (IoT) breaches, Sarma [30] outlines two steps: feature extraction and feature categorization. On this occasion, "feature extraction" refers to preparing. At this time, each program is free to develop its own set of capabilitiesIn addition, Saxe and Berlin [31] introduced a DNN for intrusion detection. With the use of multi-layer perceptron (MLPs), they developed an intrusion detection system (IDS).

By using feed-forward methods, the MLP model showcases a neural network. The research shows that deep neural networks perform much better at detecting anomalies when using rectified linear units (ReLU) as activation functions compared to other functions [31]. A similar strategy is used by Dahl et al. [34], who further improve anomaly detection by employing deep learning. They utilize a restricted Boltzmann machine (RBM), a fundamental component applicable in deep neural network (DNN) design, specifically. This might be interpreted as a pre-training method aimed at obtaining beneficial attributes to enhance accuracy.

Huang and Stokes introduced neural networks in [35] to concurrently learn many tasks. The utilisation of the neural network is the preliminary phase in assessing whether certain binary or malevolent behaviour signifies an attack. Subsequently, it was necessary to ascertain the invasion family to utilise NN. Kolosnjaji et al. [36] employed a CNN often linked with LSTM networks in their examination of invasion families. We can elucidate the interaction between the traits linked to intrusion by utilising the convolution layer.

Ullah et al. [37]'s objective was to use a Convolutional Neural Network (CNN) to store important data, and they looked at input features like the IoT. Using a mixed approach that combines conventional and deep learning methods, they were able to create a small deep learning model for binary classification. Comprehensive deep learning approach DB-CGAN, developed by Zhou et al. [38], is based on GANs [39] and uses DB. They enhanced detection and classification methodologies by acquiring robust data through adversarial training. Similarly, in the proposed deep learning model for anomaly detection This work by Kale et al. [40] used three unique methodologies: CNNs, GANomaly, and K-means clustering. This technique was predicated on artificial intelligence.

One cloud-based intrusion detection system (IDS) that Abusitta et al. [41] developed is based on deep learning. In essence, they use a Denoising Autoencoder, a more sophisticated version of the Autoencoder, to train the DNN. Denoising Autoencoders boost detection accuracy despite the presence of faulty or missing data. Consequently, even with incomplete data, the IDS may still determine possibly suspicious actions. Ultimately, Abusitta et al. [42] established an approach for proactive and collaborative malware detection.

Utilising the Denoising Autoencoder, they successfully acquired all nodes' choices, notwithstanding the incompleteness of those decisions. Consequently, their real-time classification and detection improved in accuracy [42].

A recent study proposed an alternative strategy for identifying assaults in non-stationary situations [43]. Researchers employ deep learning methodologies to extract high-level insights, enhancing attack detection in non-stationary contexts. At the core of their methodology is a denoising autoencoder, serving as a fundamental component for deep neural network training. This indicates that the system is constructed on a denoising autoencoder. Anomaly detection frameworks remain essential for IoT networks in highly unpredictable and heterogeneous situations. The rationale for this is the absence of the environment. The proposed deep learning models are often trained using high-quality data [44]. Consequently, they have difficulties in environments characterized by significant ambient noise or while processing compromised data from the Internet of Things.

This paper introduces problem detection in Internet of Things systems. By employing a combination of Gated Recurrent Units and Particle Swarm Optimization, it acquires the ability to extract resilient features. Internet of Things (IoT) systems exemplify a diverse context, and remarkably, few attributes of these systems remain unaltered by such environments. The detection of fraudulent IoT data will be particularly beneficial for techniques such as Particle Swarm Optimization, which enhance certain attributes for further use. This anomaly detection strategy for IoT models is markedly different from others that depend on deep neural networks. By separating these particulars, we can achieve enhanced objectivity.

3. Methodology

To accomplish this, we will analyze data from a real world dataset called Numenta Anomaly Benchmark (NAB) dataset, including temperature, humidity, vibration, etc. As a standard on which to measure anomaly detection system, the researchers have relied on the NAB dataset, rich in anomaly patterns that are similar to the conditions for detecting anomalous behavior on a time series. It offers a perfect testing ground to ensure the model is adaptable to different IoT scenarios and to evaluate efficiency of the alternative anomaly detection strategies.

We demonstrate for our technique to be characterized by great featurization, anomaly identification, and optimization abilities. We start by preprocessed data thoroughly by cleaning and normalising it in every step of the data. To be able to process the features uniformly, standardized data types are used by the sensors. To do so, we can remove the inconsistent data, miss value treatment, and feature scaling. For consistent feature distributions on the data, we need to do data normalisation to keep the same behaviour of the data on the data, to improve the anomaly detection model performance.

It performs feature extraction and preservation of most relevant properties and reduces dimensionality, which results into high input data quality. A few machine learning and statistical approaches to identifying and keeping the most useful features include the use of correlation analysis, variance thresholding, mutual information analysis, and Fisher score ranking. Using this we choose some characteristics that are statistically significant as well as computationally efficient that will help us discard any unnecessary or noisy data which may corrupt the model's accuracy.

We then start to group the data points based on their similarity in terms of k application through sophisticated clustering algorithms after data pretreatment and feature selection. So here, we are going to consider the implementation of K Means, DB Scan, and Mean Shift, three clustering algorithms, and then identify which one may do the job well in terms of clustering anomalous outliers. Our results suggest that the best approach in distinguishing between typical and out of the ordinary data distributions is K-Means clustering.

This technique is very dependent on a Gated Recurrent Units (GRU) based anomaly detection model. GRU, a Recurrent Neural Network (RNN), which enjoys excellent suitability for analyzing sequential data, is an incredible choice when it comes to detecting IoT anomalies. Time series sensor data necessary for training is used to learn temporal dependence and pattern detection related to the abnormalities. We further optimize performance of the GRU model using learning rate, hyper parameters, batch size and number of hidden units, by using Particle Swarm Optimization (PSO). This is useful for tuning hyper parameters, and PSO achieves this by allowing dynamic changing of hyper parameters, changing the model correctness.

We carry out extensive tests to check the efficacy of the method suggested, using the performance indicators, such as accurate, precisions, recalls, F1 score, etc. For example, AUC-ROC curves provide some illustrations. In addition, the model is also evaluated against standard ML classifiers, such as K-Nearest Neighbours (KNN), Random Forest, Naïve Bayes, Decision Trees etc, and some of the most recent and advanced anomaly detection methods. In this work, we present an approach for building the high performance anomaly detection framework to enable high speed and real time detection of the suspicious activity and system irregularity, while confidently, and securely rejecting all other activity in the IoT environment.

Pre-processing:

1. By Data cleaning: We will do basic data cleaning techniques to make our data clean and ready to process. This is implementing such that it will first check for missing values and drop columns that don't have sensor data.
2. Feature Selection/Extraction: The features that we will pick out from the dataset such that they contain sensor data will be using methods such as independent component analysis (ICA) and correlation based feature selection (CFS). It will aid in lowering noise as well as extracting the main features of the data.
3. DBScan clustering algorithm will be used to cluster data points if they are close in terms of their feature values. It will help to detect any clusters that can be thought of as representing the accelerometer value being anomalous.
4. Label-Encoding: The method using which we shall be performing normalization, in that the attributes shall be scaled to give out similar value ranges. It will improve the convergence and performance of the model.
5. Prior to addressing the curse of dimensionality, techniques like Principal Component Analysis (PCA) or t-SNE will be used to decrease the data's dimensionality. As a result, the model's speed and performance will be enhanced.
6. Outlier Data Processing: For outlier data processing we will use the techniques like Z-score, Local Outlier Factor (LOF), Isolation Forest etc to detect and delete outliers from the dataset. This will allow for improvement of the model performance as well as accuracy.

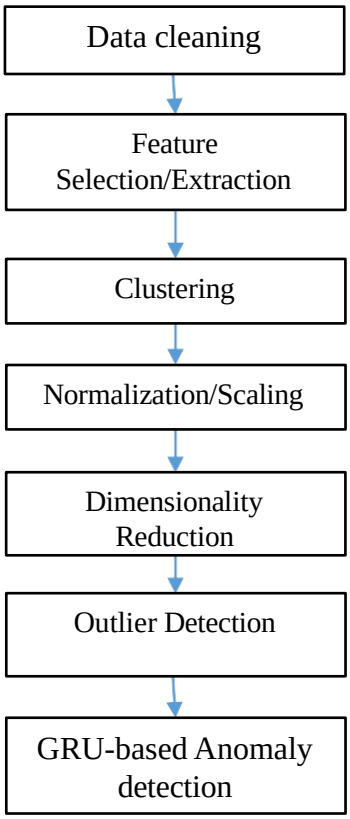
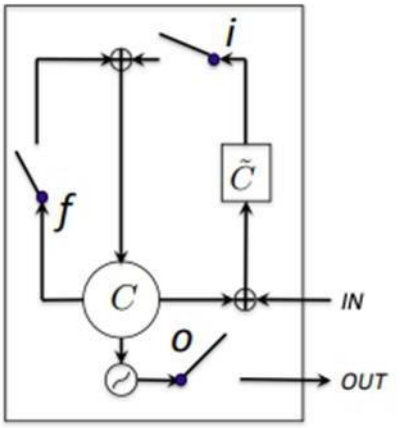


Figure 3.1: Flowchart of the proposed research

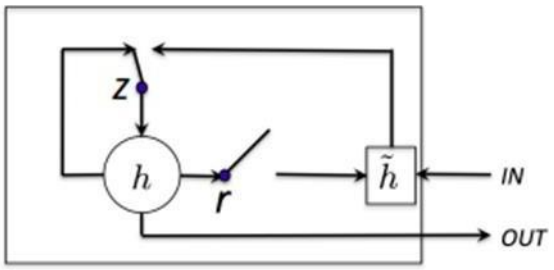
3.1 Gated Recurrent Unit

The Gated Recurrent Unit is a variation of the LSTM that stands to provide some simplification with respect to its predecessor. Additional to the 'update gate' made by the combination of the 'forget' and input gates there is a 'reset gate'. Finally, the last variant is easy to understand and is being increasingly recognized as popular these days.

However, unlike an LSTM, a Gated Recurrent Unit does not require a separate memory cell in order to modulate information contained within the unit. Both GRU and LSTM share many characteristics in common, however these two models also have some key distinctions.



(a) Long Short Term Memory Unit. i: Input Gate, f: Forget Gate, o: Output Gate.



(b) Gated Recurrent Unit. z: update Gate, r: Reset Gate.

Figure 3.2 : Gated Recurrent Unit

3.2 Particle Swarm Optimization

Among the many heuristic strategies used to solve complex and analytically difficult problems in recent years, PSO has emerged as a crucial one. This category includes issues such as electrical power systems, data mining information extraction, work scheduling, and similar categories of problems. I find it fascinating how species in general exhibit social behaviour, like a school of fish or a flock of birds. For a rundown of the key points that make up the PSO, see this list [14]:

In order to find the optimal solution, a random generator generates a set of all potential solutions, which is called the initial population. What we call the solutions that are present in the population are particles.

A particle's output is determined by its fitness value, according to the fitness function. It finds the average population fit of an existing person (a solution).

Selection: In each cycle, two elements are utilised to ascertain the next location for each particle. Each particle's location is defined by these characteristics. These qualities are known as the personal best and the global best respectively. One describes the optimal location that a particle has reached on its own during exploration, while the other describes the optimal location relative to all other particles.

Updation: After determining these two most accurate values, the following equation is used to do an update on the velocity and position of a particle:

$$Vnew = Vold + c1\,X\,rand()X(pbest - present) + c2\,X\,rand()X(gbest - present)$$
$$Present\ new = present\ old + Vnew$$

where rand() denotes a random value among 0 and 1, c_1 and c_2 represent the learning factors, and V is denotes the particle velocity.

4. Experimental results

4.1 Dataset

The dataset utilized for the present research is the CICDarknet2020 dataset. At the first tier of the CICDarknet2020 dataset, a two-layered method is utilized so as to achieve both darknet and benign traffic. Browsing, VOIP, Video-Stream, Transfer, P2P, Email, Chat, and Audio-Stream are the components that make up the dark web traffic. This traffic is generated on the second layer. In order to produce a dataset that is indicative of the whole, two datasets are combined that had been previously produced, namely ISCXTor2016 and ISCXVPN2016. We then integrated the traffic from Tor and VPN services into categories that correspond to Darknet activity.

Link: https://www.kaggle.com/datasets/peterfriedrich1/cicdarknet2020-internet-traffic

The sample of the dataset is given in figure 4.1. The shape of the dataset is (873611, 79).

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min	Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s	Flow Packets/s	Flow IAT Mean
170361	55641	49	1	3	6	18	6	6	6.000000	0.000000	6	6	6.000000	0.000000	489795.918400	81632.653060	16.333333
170362	45337	217	2	1	31	6	31	0	15.500000	21.920310	6	6	6.000000	0.000000	170506.912400	13824.884790	108.500000
170363	22	1387547	41	46	2728	6634	456	0	66.536585	110.129945	976	0	144.217391	307.913979	6747.158835	62.700579	16134.267440
170364	22	207	1	1	0	0	0	0	0.000000	0.000000	0	0	0.000000	0.000000	0.000000	9661.835749	207.000000
170365	60146	50	1	2	0	0	0	0	0.000000	0.000000	0	0	0.000000	0.000000	0.000000	60000.000000	25.000000

Figure 4.1 : Sample dataset

4.2 Pre-processing steps

The collection contains several forms of assaults. BENIGN, Bot, DDoS, PortScan, Brute Force, Web Attack, and XSS are the ones listed. In figure 4.2, we can see the label distribution.

```
array(['BENIGN', 'DDoS', 'PortScan', 'Bot', 'Web Attack \x96 Brute Force',
       'Web Attack \x96 XSS', 'Web Attack \x96 Sql Injection'], dtype=object)
```

Label	
PortScan	35557
DDoS	5953
BENIGN	3421
Bot	1825
Web Attack ◆ Sql Injection	1544
Web Attack Brute Force	1502
DoS GoldenEye	8

Name: count, dtype: int64

Figure 4.2 : Label Distribution

As a pre-processing step, data Cleaning is done by duplication removal and null data removal, Infinite and hash value removal on the dataset. Shape After Data Cleaning is (44276, 79). Label Distribution After Data Cleaning is given in figure 4.3.

Label	
PortScan	30074
DDoS	5953
BENIGN	3400
Bot	1819
Web Attack ◆ Sql Injection	1532
Web Attack Brute Force	1490

Name: count, dtype: int64

Figure 4.3 : Label Distribution After Data Cleaning

Label Distribution Chart

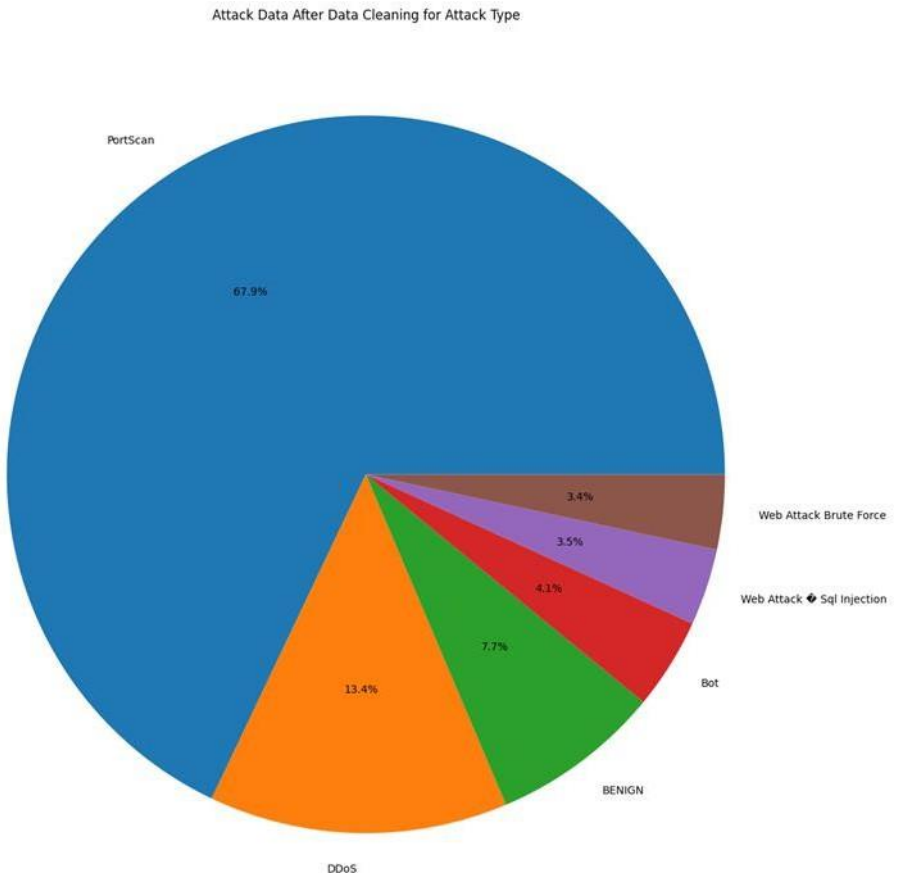
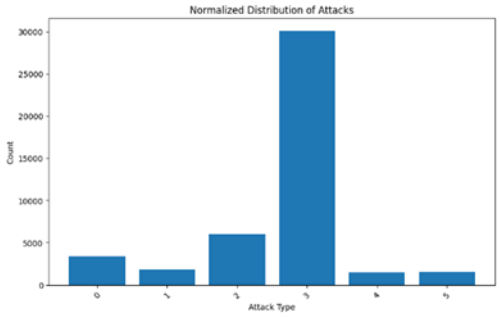


Figure 4.4 : Label Distribution Chart

4.3 Normalization

Among the many data preparation strategies that are utilized in machine learning, normalization is one of the most common. When all of the columns in a dataset have their sizes adjusted to be uniform, this procedure is called normalization. The suggested study use Label Encoder as its normalization method. Label encoding simplifies and expedites the process of numerically representing category data. Encoded Label Distribution is given in figure 4.5. Dataset After Normalization is given in figure 4.6.




```
array([4, 2, 5, 0, 3, 1])
```

Figure 4.5 : Encoded Label Distribution

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min	Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s	Flow Packets/s	Flow IAT Mean
0	39916	3	2	0	31	0	31	0	15.500000	21.920311	0	0	0.000000	0.000000	1.030000e+07	666666.687500	3.000
1	80	9327760	5	0	30	0	6	6	6.000000	0.000000	0	0	0.000000	0.000000	3.216206e+00	0.536034	2331940.000
2	80	143121	3	6	26	11607	20	0	8.666667	10.263203	4380	0	1934.500000	2177.344971	8.128088e+04	62.883854	17890.125
3	80	713412	3	5	26	11607	20	0	8.666667	10.263203	7215	0	2321.399902	3327.769775	1.630615e+04	11.213717	101916.000
4	80	1826338	3	6	26	11601	20	0	8.666667	10.263203	4380	0	1933.500000	1757.789917	6.366292e+03	4.927894	228292.250

Figure 4.6: Dataset after Normalization

4.4 Feature Selection and Extraction

Feature extraction and feature selection follow normalization. Classification of mutual information is carried out using the procedures of variance threshold, correlation coefficient, and fisher score. To exclude characteristics with low variance that aren't relevant for modelling, a feature selection approach called the variance threshold is used to a dataset. To do this, the method is applied to the dataset. Learning without supervision is possible because it zeroes focused on the features—the inputs—rather than the outputs the outcomes. The Threshold variable is initially set to 0 by default. After that, we'll figure out the non-constraint values and the constraint values. Features unrelated to content number 69.

Both the training and testing data sets are considered using the Correlation Confusion Matrix heat map. In order to examine the level of association between specific

quantitative variables, it is common practice to compute correlation coefficients. Finding a connection between the variables is what this study is all about. A rise in one measure is indicative of an increase in the other, and this relationship is known as a positive correlation. It is said that two variables have a negative correlation when high values of one are found to go hand in hand with low values of the other. When the two variables' values are same, however, we say that there is a positive correlation between them. We have 45 characteristics for correlation. In figure 4.7, you can see the heatmap of the correlation coefficient.

By reducing Fisher's score—which is the derivative or gradient of the log likelihood function—to zero, we obtain the most probable parameter estimate. Because it is a gradient function, the log likelihood function explains this. In figure 4.8, you can see the Fisher score graph.

Following correlation, we examine mutual information classification. The correlation coefficient in the Fisher score. The value of mutual information (MI), which quantifies the degree to which two random variables are dependent on one another, is positive. If the value is 0, then the two random variables are independent; if it's larger, then the relationship is stronger. A graph displaying the M.I. scores for each column is provided after considering Mutual Information Classification for the target Label and Data Attribute. Figure 4.9 displays the M.I. score for each column. The final form following feature extraction and selection is (44227, 24). Figure 4.10 displays the data after feature extraction and selection.

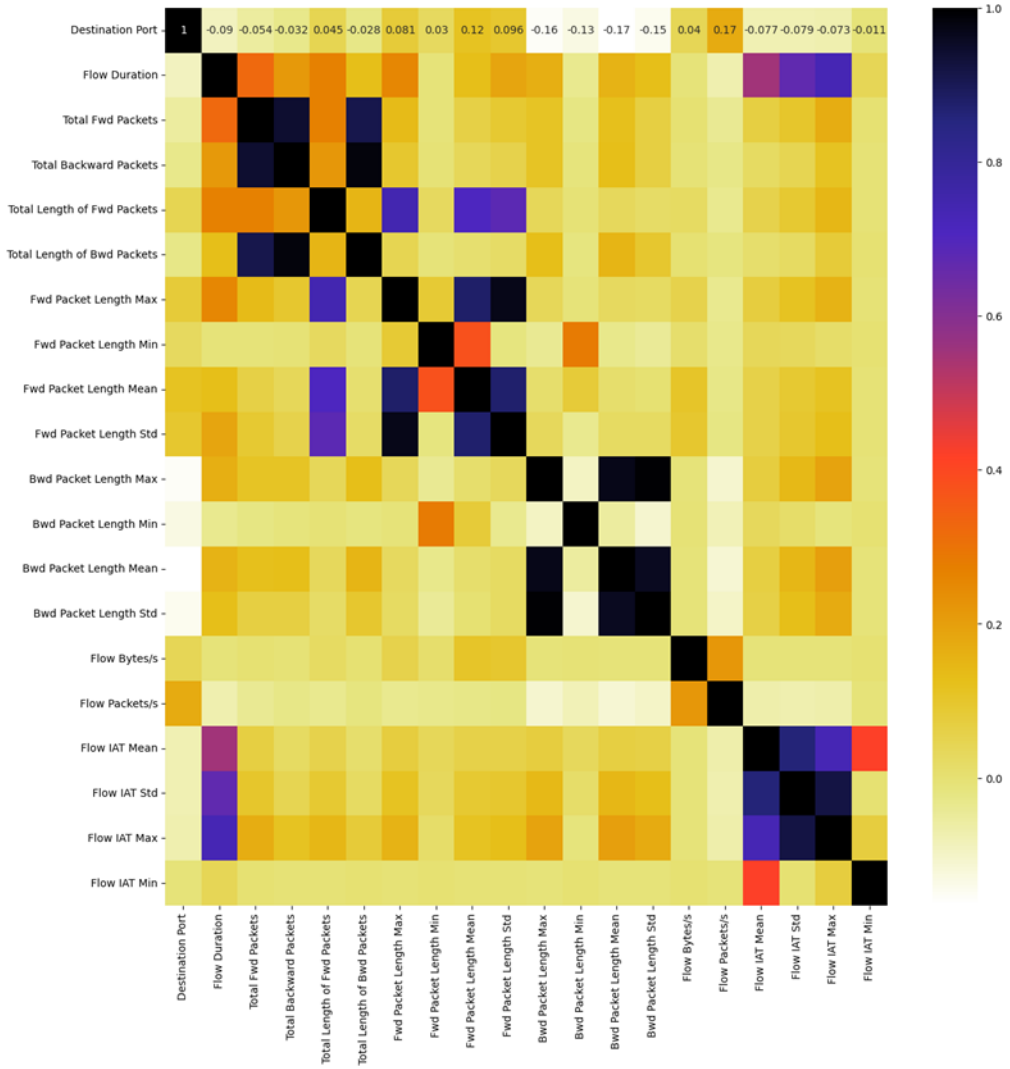


Figure 4.7 : Correlation Coefficient

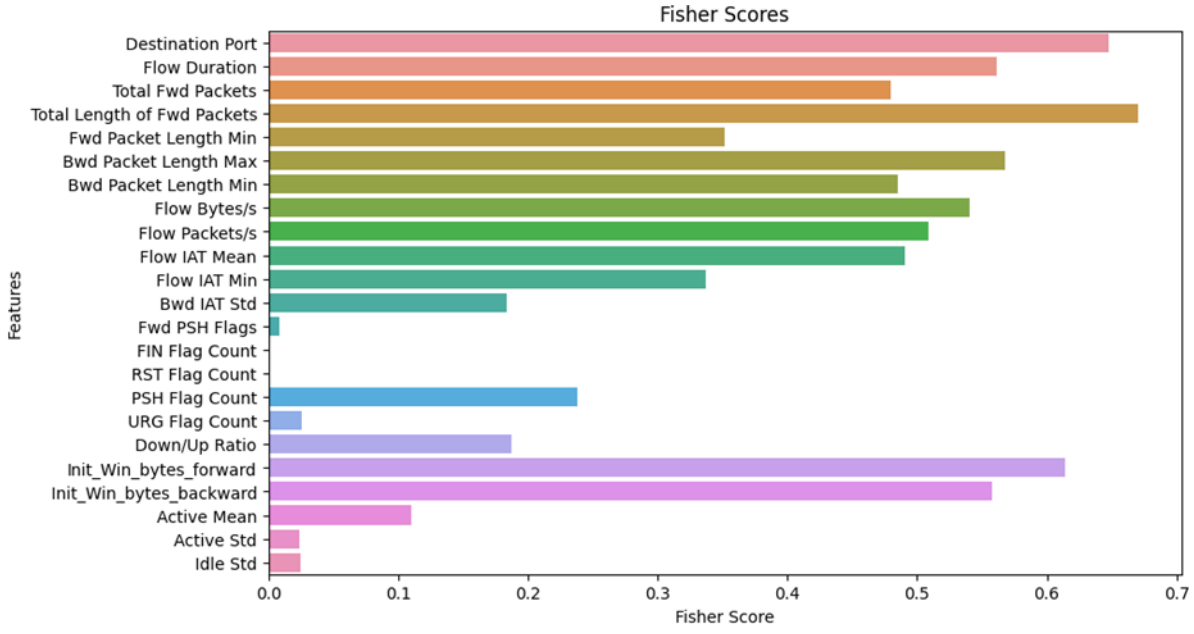


Figure 4.8 : Fisher score.

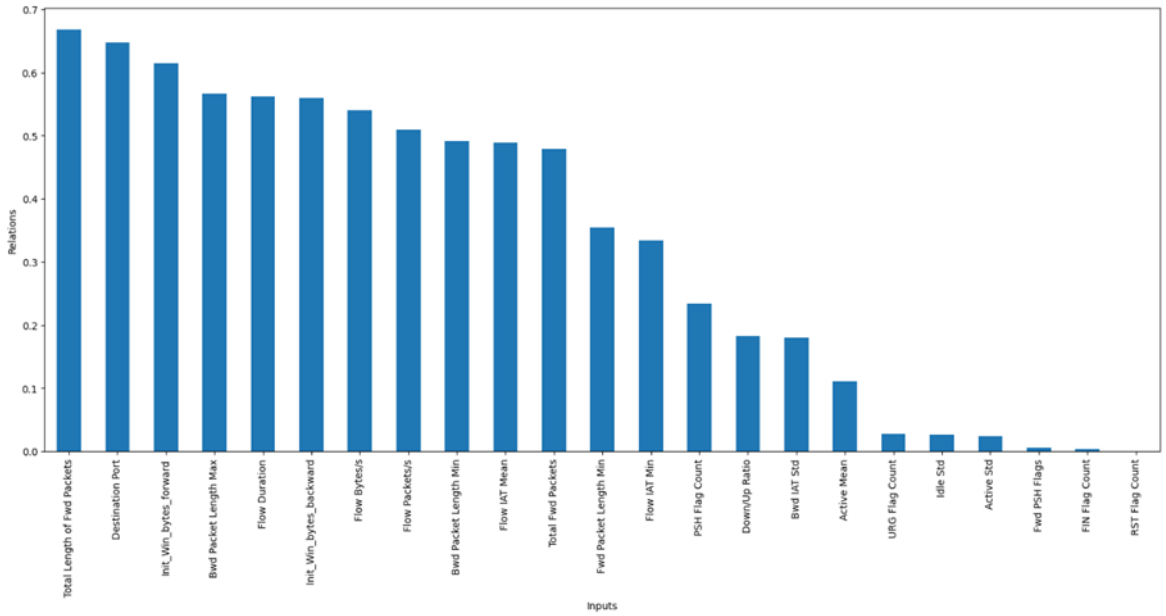


Figure 4.9 : Mutual Information Classification

	Destination Port	Flow Duration	Total Fwd Packets	Total Length of Fwd Packets	Fwd Packet Length Min	Fwd Packet Length Max	Bwd Packet Length Min	Flow Bytes/s	Flow Packets/s	Flow IAT Mean	Flow IAT Min	Bwd IAT Std	Fwd PSH Flags	FIN Flag Count	RST Flag Count	PSH Flag Count	URG Flag Count	Down/Up Ratio	Init_Win_by
0	39916	3	2	31	0	0	0	1.030000e+07	666666.687500	3.000	3	0.000000	1	0	0	0	0	0	
1	80	9327760	5	30	6	0	0	3.216206e+00	0.536034	2331940.000	1	0.000000	0	0	0	0	0	0	
2	80	143121	3	26	0	4380	0	8.128088e+04	62.883854	17890.125	16	62734.714844	0	0	0	1	0	2	
3	80	713412	3	26	0	7215	0	1.630615e+04	11.213717	101916.000	10	356007.687500	0	0	0	1	0	1	
4	80	1826338	3	26	0	4380	0	6.366292e+03	4.927894	228292.250	64	811656.250000	0	0	0	1	0	2	

Figure 4.10 : Data after feature selection and extraction

4.5 Clustering and Anomaly Detection

For the proposed research, clustering is used for detecting the anomaly. Various clustering techniques such as K-Means, DBScan and Mean shift clustering is used for comparing the best clustering option. K-means is selected for its higher performance values. The comparison of the clustering scores are given in table 4.1. Clustering Comparison is given in figure 4.11.

Table 4.1 : Clustering Comparison

Clustering technique	Silhouette	Davies bouldin	Calinski harabasz
K-means	0.965	0.312	38759.322
DBSCAN	0.419	2.408	798.259
Mean shift	0.910	0.356	11093.303

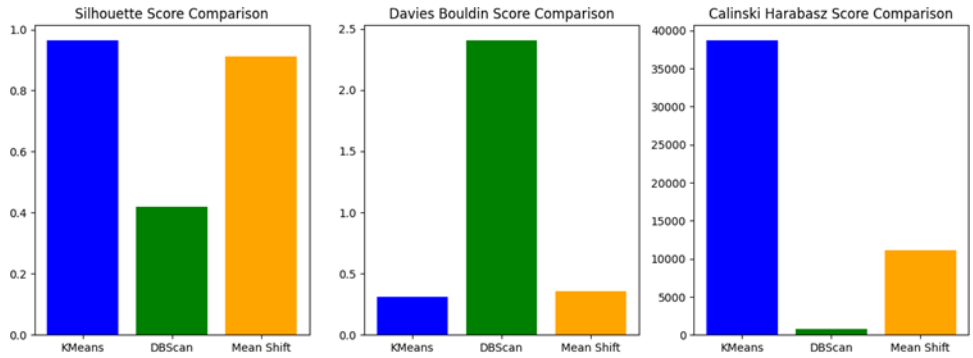


Figure 4.11 : Clustering Comparison

Finding Abnormalities K-Means based on outliers. Systemic problems include erroneous results and models, skewed parameter estimations, and outliers might be symptoms of more widespread abnormalities in the data. Prior to proceeding with modelling and analysis, outlier identification allows one to identify these unusual data patterns. In order to find anomalies in networks, several outlier detection approaches have been created and used. In order to discover anomalies, K-Means Cluster Label Creation is carried out.

Z-Score for Outlier Selection is taken into account when the outliers are analyzed. One way to quantify the dispersion of a set of results around the mean is via the Z-score, a statistical metric.

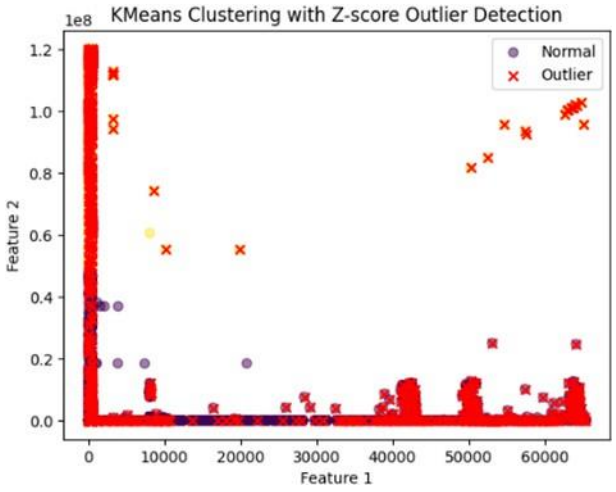


Figure 4.12 : K-Means clustering with Z-score outlier detection

Setting E-Distance Threshold using Local Outlier Factor is done next. An algorithm known as local outlier factor (LOF) is used to determine the outliers that are contained inside a dataset. When the population density of the area is taken into consideration,

LOF will locate an anomaly. When the data density does not remain consistent throughout the collection, LOF works very well. Setting Anomaly without K-Means, the LOF anomaly detection is given in figure 4.13.

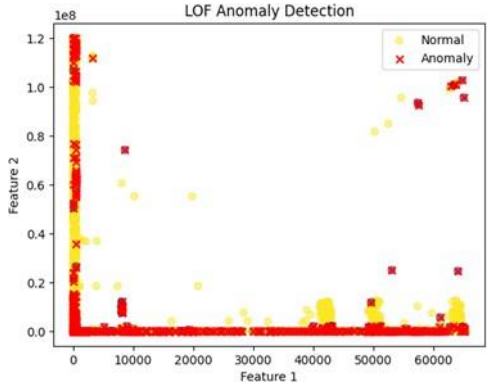


Figure 4.13 : LOF anomaly detection without K-Means

Isolation Forrest training is done on the dataset. The above procedure continues until all of the insights have been "isolated." The final Anomaly Detection is given in figure 4.14. Cluster and Anomaly Column are later added.

```
array([-1,  1,  1, ...,  1,  1,  1])
```

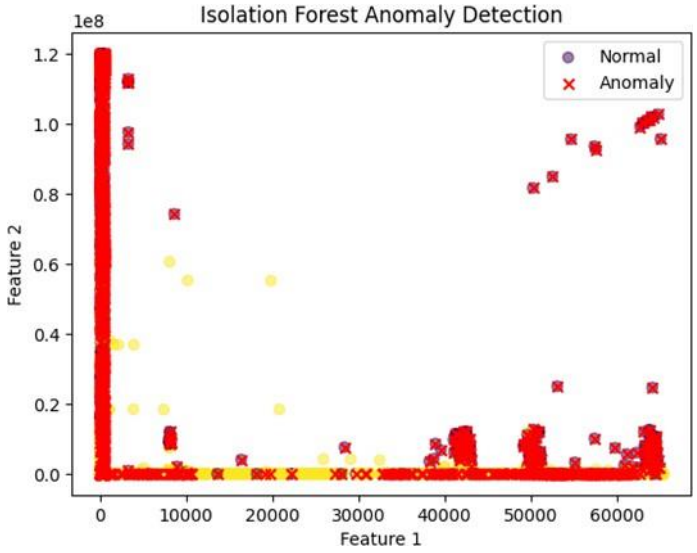


Figure 4.14 : Final anomaly detection

4.6 PCA Analysis and Dimension Reduction

Principal Component Analysis (PCA) is of a very important unsupervised learning technique used in machine learning to deal with problem of dimensionality reduction. PCA reduces the dimensionality of extensive datasets using a systematic process that

reduces dimensionality and increases computational efficiency as well as simplifies data visualization and interpretation. This statistical procedure performs orthogonal transformation to transform original correlated variables into new set of uncorrelated features called principal components. They are the components of data on the axes along which the variance is maximized and describe the highest value information contained in this dataset. PCA achieves this reduction through minimal loss of information and enhances a highly correlated set of attributes to achieve a much more simplified and redundant free version of the original set. This process defines eigenvectors and eigenvalues of the covariance matrix with the help of which the mapping is done onto orthogonal coordinates.

The PCA applies this logic by identifying the principal components based on their eigenvalues and retaining only the ones with highest eigenvalue which together explain the majority of the variance in the dataset and simplifies the dataset without losing its key characteristics. In doing so, it reduces the dimensions in a selective manner, thus improving the results of subsequent analyses (e.g. clustering, classification, etc) such as predictive modelling. Furthermore, PCA is very useful for visualization, since it allows to visualize complex, high dimensional data in at least 2 or 3 dimensions.

Figure 4.15 gives an illustrative depiction of PCA analysis and its dimensionality reduction process.

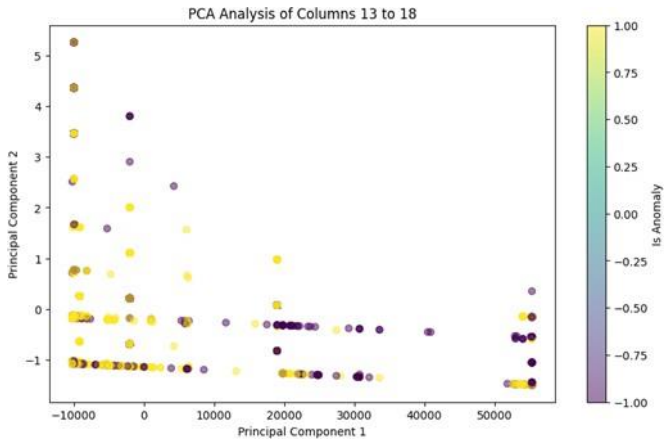


Figure 4.15 : PCA analysis and dimension reduction

The dataset After Clustering, Anomaly Detection and PCA is given in figure 4.16.

	Destination Port	Flow Duration	Total Fwd Packets	Total Length of Fwd Packets	Fwd Packet Length Min	Bwd Packet Length Max	Bwd Packet Length Min	Flow Bytes/s	Flow Packets/s	Flow IAT Mean	Flow IAT Min	Bwd IAT Std	Fwd PSH Flags	FIN Flag Count	RST Flag Count	PSH Flag Count	URG Flag Count	Down/Up Ratio	Init_Win_b
0	39916	3	2	31	0	0	0	1.030000e+07	666666.687500	3.000	3	0.000000	1	0	0	0	0	0	
1	80	9327760	5	30	6	0	0	3.216206e+00	0.536034	2331940.000	1	0.000000	0	0	0	0	0	0	
2	80	143121	3	26	0	4380	0	8.128088e+04	62.883854	17890.125	16	62734.714844	0	0	0	1	0	2	
3	80	713412	3	26	0	7215	0	1.630615e+04	11.213717	101916.000	10	356007.687500	0	0	0	1	0	1	
4	80	1826338	3	26	0	4380	0	6.366292e+03	4.927894	228292.250	64	811656.250000	0	0	0	1	0	2	

Figure 4.16 : Dataset After Clustering, Anomaly Detection and PCA

4.7 Training and Testing of dataset

The dataset is split into the train and test dataset. The ratio in which the dataset is split is ((33170, 29), (11057, 29), (33170,), (11057,)). The Train Test Distribution Graph is given in figure 4.17.

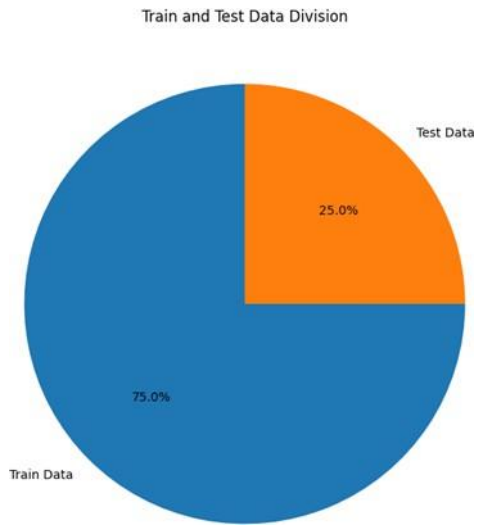


Figure 4.17 : Train Test Distribution Graph

4.8 PSO Optimizer for Hyperparameter Tuning

PSO is configured on the dataset and the optimizer is used for hyperparameter tuning after training and testing of the dataset. PSO Execution takes place and PSO Graphs are generated. PSO Best Optimum Position (for Epoch and Batch Size of GRU) is (23, 8). Runtime chart for the PSO optimizer is given in figure 4.18. Exploration vs exploitation graph is given in figure 4.19. The standardization takes place after optimization.

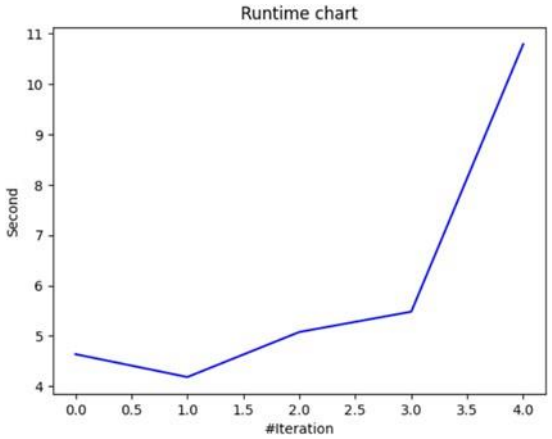


Figure 4.18 : Runtime chart

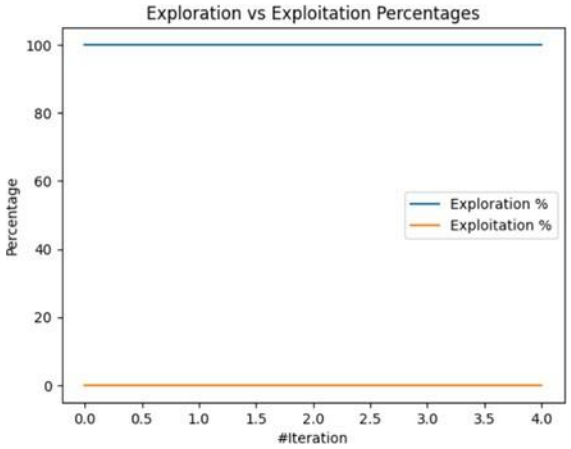


Figure 4.19 : Exploration vs exploitation graphs

4.9. Gated Recurrent Unit Model

Using gating methods to selectively update the hidden state of the network at each time step is the core concept of GRU, which was developed by Google Research Underground. Controlling the flow of information into and out of the network is accomplished through the utilization of the gating mechanisms. The GRU is equipped with two different kinds of gates, which are referred to as the reset gate and the update gate. Model Summary of the GRU model is given in figure 4.20. Model's Accuracy to Loss Ratio Graph is represented in figure 4.21.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_12 (GRU)                (None, 29, 64)            12864

 dropout_12 (Dropout)        (None, 29, 64)            0

 gru_13 (GRU)                (None, 29, 64)            24960

 dropout_13 (Dropout)        (None, 29, 64)            0

 gru_14 (GRU)                (None, 29, 64)            24960

 dropout_14 (Dropout)        (None, 29, 64)            0

 gru_15 (GRU)                (None, 64)                24960

 dropout_15 (Dropout)        (None, 64)                0

 dense_3 (Dense)             (None, 6)                 390

=================================================================
Total params: 88134 (344.27 KB)
Trainable params: 88134 (344.27 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.20 : GRU model

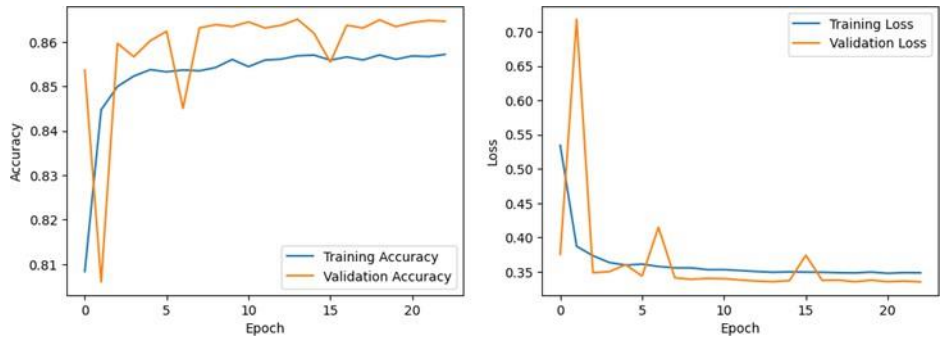


Figure 4.21 : Model's Accuracy to Loss Ratio Graph

4.10 Testing and Evaluation

In machine learning, the majority of our attention is directed towards model assessment, which consists of metric calculations and charts that summarize the accuracy of a model on an unknown holdout test data set. Testing a model, on the other hand, involves determining whether or not the learnt behaviour of the model is consistent with "what we expect." It is not defined with the same level of rigour as model evaluation. The prediction and Performance Matrices for the Gated Recurrent Unit and Particle Swarm Optimization is

```
Accuracy:  0.8805408338609026
Precision:  0.8744523268735394
Recall:  0.8605408338609026
F1 Score:  0.8467645221699409
```

As shown in Fig. 4.22, the performance metrics of the proposed model are represented in a graphical form. The visual analysis of this gives a clear insight as to how the model performs on different key indicators and thus it becomes easier to interpret the model results. Additionally, Figure 4.23 shows the confusion matrix, an important tool of evaluation of classification accuracy and model performance of the proposed

model. It also gives complete picture of how true positives, true negatives, false positives and false negative would look like in a confusion matrix.

Figure 4.22: Visual depiction of the proposed model's performance metrics.

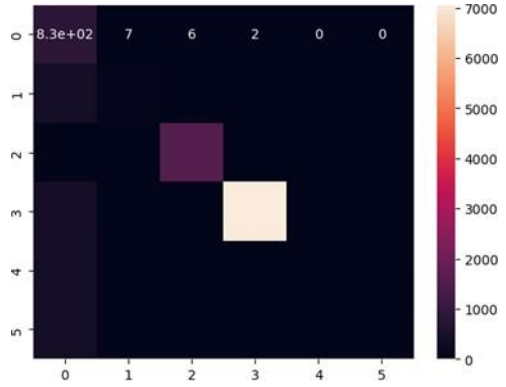


Figure 4.23 : Confusion matrix of the proposed model

This model takes the suggested GRU+PSO error rates into account. When calculating the mistakes, R2 score, Mean Squared Error, Mean Absolute Error, and Mean Pinball Loss are taken into account. In figure 4.24, we can see the suggested model's error rates graphically shown.

```
Mean Squared Error:  1.3220977573073032
Mean Absolute Error:  0.44994121371077145
R2 Score:  -0.684469016355896
Mean Pinball Loss:  0.40292122637243377
```

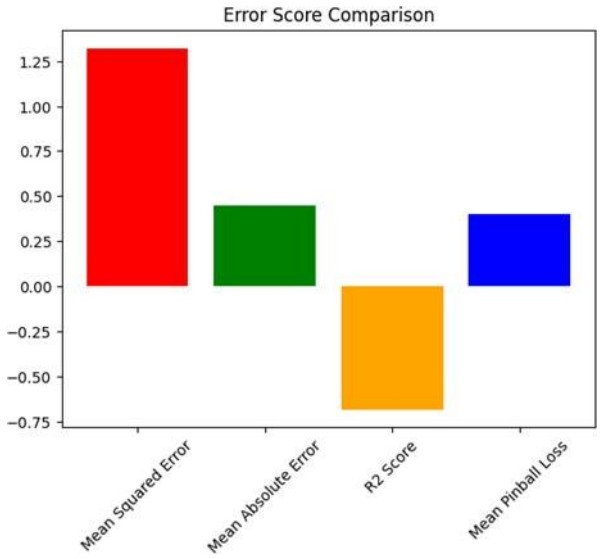


Figure 4.24 : The error rates of the proposed model

The Area Under the ROC Curve, known as AUC ROC, has been used as a robust measure for assessing the performance of any machine learning algorithm specifically the binary classifiers. The ROC curve itself is a plot that labels the true positive rate (sensitivity) vs. the false positive rate (1-specificity) vs. different decision thresholds.

In practice, ROC curve is how well a binary classification model can truly differentiate between two classes. The ROC curve is a plot of each pair of sensitivity and specificity for a given decision threshold. Furthermore, a classifier that possesses high discriminatory power will have a plot close to the top left corner of the plot, where the true positive rate is relatively high and the false positive rate is relatively low. An alternative form of a classifier that does no better than chance is a line on the diagonal from the bottom left corner to the top right corner.

The ROC curve is used to succinctly quantify the overall performance represented by the curve by computing the metric called Area Under the Curve (AUC). An AUC value of 0 means the model doesn’t perform well, while a value that is closer to 1 means the more the model can perform and be more reliable. In particular, a perfect AUC closest to 1 means excellent discriminative ability, a moderate AUC closest to 0.5 is equivalent to random guessing and a bad AUC far less than 0.5 indicates that the model behaves the opposite of the expectation.

Graphical illustration is given to the ROC curve and of the corresponding AUC in Figure 4.25 explicit in showing one's contempt in the classifier in being able to discriminate classes.

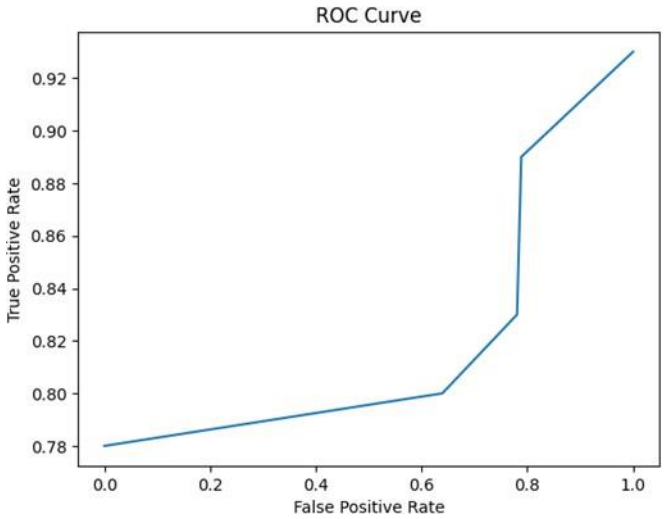


Figure 4.25 : AUC-ROC characteristic curve

4.11 Comparison of proposed model with GRU without PSO and Extra Layers

In above Gated Recurrent Unit without PSO and having extra layers. Extra layers are added to the GRU model and considered for model training. The model summary of our planned methodology seems in 4.26. Which is determining that graphical representation is given in figure 4.27. The Confusion Matrix of the comparison model is given in figure 4.28.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_16 (GRU)                (None, 64)                12864

 dense_4 (Dense)             (None, 6)                 390

=================================================================
Total params: 13254 (51.77 KB)
Trainable params: 13254 (51.77 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.26 : Model summary

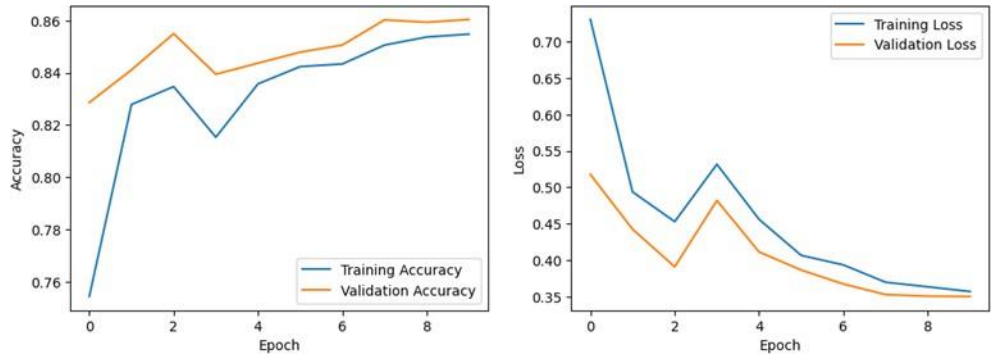


Figure 4.27 : Accuracy to Loss Ratio

The prediction and Scores of the GRU without PSO having extra layers

```
346/346 [==============================] - 15s 43ms/step
Accuracy : 0.81684324562
Precision : 0.8276515514
Recall : 0.813586526355
F1 : 0.80452458254524
```

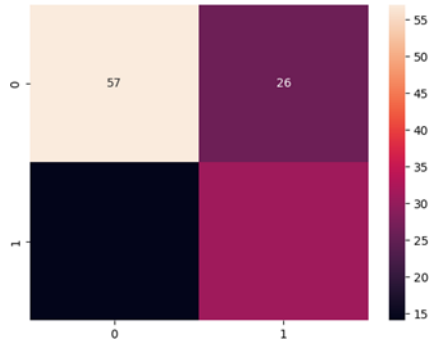


Figure 4.28 : Confusion Matrix of the comparison model

The proposed model is compared with the KNN, Random Forest, Naive Bayes, Decision Tree, Extra Tree and AdaBoost classifiers. The performance scores classifiers are given in table 4.2. Confusion matrix of (a) KNN (b) Random Forest (c) Naive Bayes (d) Decision tree (e) Extra Tree and (f) Adaboost

Table 4.2 : Comparison of performance metrics of the comparison classifiers

Algorithm	Accuracy	Precision	Recall	F1-score
KNN	0.8330	0.8257	0.8330	0.8268
Random Forest	0.8404	0.8405	0.8404	0.8391
Naive Bayes	0.7061	0.6858	0.7061	0.6717
Decision Tree	0.8307	0.8266	0.8307	0.8285
Extra Tree	0.8373	0.8356	0.8373	0.8358
Adaboost	0.8125	0.7673	0.8125	0.7855

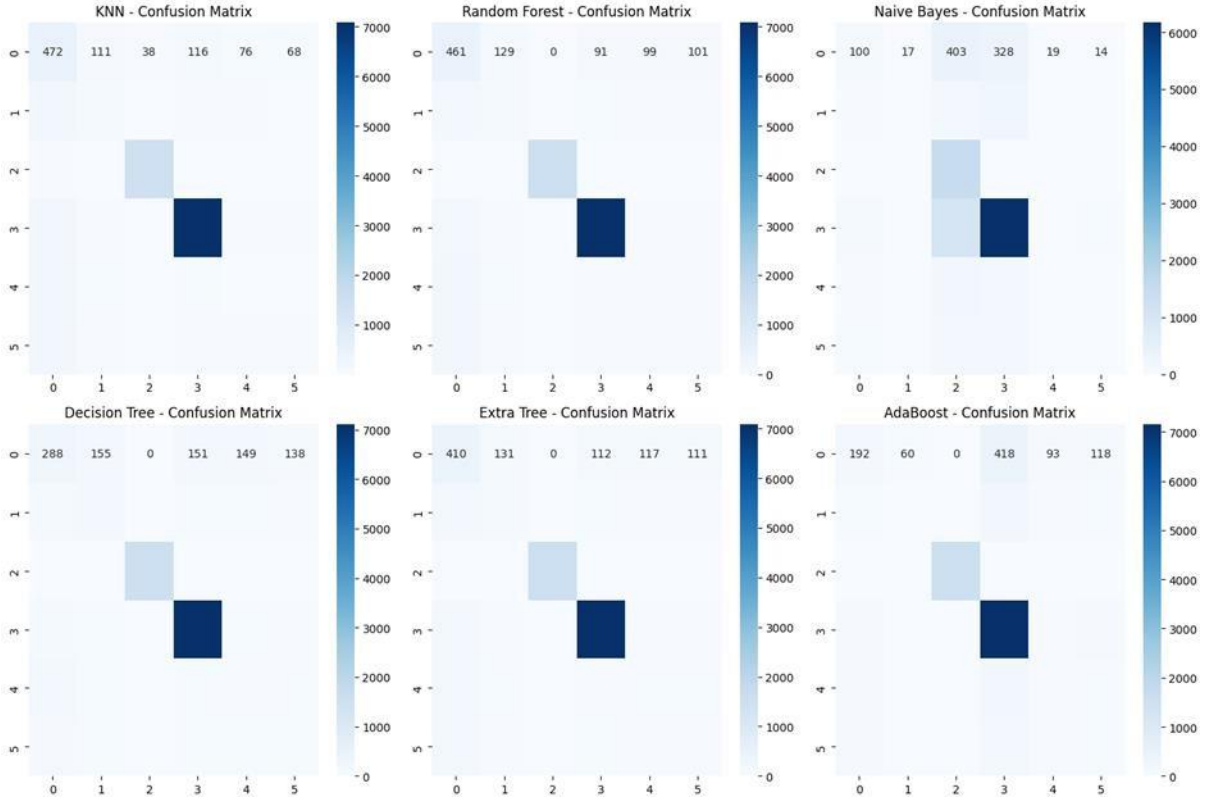


Figure 4.29 : Confusion matrix of (a) KNN (b) Random Forest (c) Naive BaConfusion matrix of (a) KNN (b) Random Forest (c) Naive Bayes (d) Decision tree (e) Extra Tree and (f) Adaboost

5. Conclusion

A deep learning-based solution was integrated with Gated Recurrent Unit (GRU) and Particle Swarm Optimization (PSO) in order to improve the accuracy and reliability of anomaly detection in Internet of Things (IoT) environments. The use of a mix of clustering methods, feature extraction, and deep learning is one of the suggested solutions to the problem of real-time anomaly and malicious activity detection. In conclusion, using a GRU-based anomaly detection framework, ES outperforms typical machine learning classifiers. It does this by utilizing PSO for hyper parameter optimization, which allows it to achieve superior performance.

The suggested GRU-PSO model outperforms conventional anomaly detection methods in these types of issues in terms of detection precision, recall accuracy and F1 score, all while using the same amount of CPU resources. Our approach is validated by comparing the results achieved to benchmark counterparts of IoT

security threat detection such as Random Forest, K Nearest Neighbors (KNN), Naive Bayes, and the Decision Trees. The practical application of the model to real world IoT scenarios which involves heterogenous time series and anomalous patterns is evidenced by the model's capability to handle such data, including its anomaly patterns.

Future direction for improving the model robustness will include replacing Constants and EventLog models with genetic algorithm for and reinforcement learning optimization, thereby increasing anomaly detection accuracy. Under distributed and federated learning paradigms, extending the framework would also enable a more scalable and adaptive security solution which is independent of a centralized data processing. The emerging cyber threats surrounding IoT security means that IoT security solutions will continue to evolve in order to mitigate them, increase the integrity and reliability of those systems.

References

[1] I. Makhdoom, M. Abolhasan, J. Lipman, R.P. Liu, W. Ni, Anatomy of threats to the internet of things, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1636–1675.

[2] I. Cvitić, D. Peraković, M. Periša, M. Botica, Novel approach for detection of IoT generated DDoS traffic, Wirel. Netw. 27 (3) (2021) 1573–1586.

[3] Y.-Q. Chen, B. Zhou, M. Zhang, C.-M. Chen, Using IoT technology for computer-integrated manufacturing systems in the semiconductor industry, Appl. Soft Comput. 89 (2020) 106065.

[4] Y. Tan, W. Yang, K. Yoshida, S. Takakuwa, Application of IoT-aided simulation to manufacturing systems in cyber-physical system, Machines 7 (1) (2019) 2.

[5] O.A. Wahab, Intrusion detection in the IoT under data and concept drifts: Online deep learning approach, IEEE Internet Things J. (2022).

[6] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N.O. Tippenhauer, H. Sandberg, R. Candell, A survey of physics-based attack detection in cyber-physical systems, ACM Comput. Surv. 51 (4) (2018) 1–36.

[7] J.E. Rubio, C. Alcaraz, J. Lopez, Preventing advanced persistent threats in complex control networks, in: European Symposium on Research in Computer Security, Springer, 2017, pp. 402–418.

[8] O.A. Wahab, J. Bentahar, H. Otrok, A. Mourad, How to distribute the detection load among virtual machines to maximize the detection of distributed attacks in the

cloud? in: 2016 IEEE International Conference on Services Computing (SCC), IEEE, 2016, pp. 316–323.

[9] A.-R. Sadeghi, C. Wachsmann, M. Waidner, Security and privacy challenges in industrial internet of things, in: 2015 52nd ACM/EDAC/IEEE Design Automation Conference, DAC, IEEE, 2015, pp. 1–6.

[10] M.A. Al-Garadi, A. Mohamed, A.K. Al-Ali, X. Du, I. Ali, M. Guizani, A survey of machine and deep learning methods for internet of things (IoT) security, IEEE Commun. Surv. Tutor. 22 (3) (2020) 1646–1685.

[11] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, H. Ming, Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning, in: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC, IEEE, 2019, pp. 0305–0310.

[12] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, Internet Things 7 (2019) 100059.

[13] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, 2019, arXiv preprint arXiv:1901.03407

[14] P. Ducange, G. Mannará, F. Marcelloni, R. Pecori, and M. Vecchio, "A novel approach for internet traffic classification based on multi-objective evolutionary fuzzy classifiers," in *2017 IEEE international conference on fuzzy systems (FUZZ-IEEE)*, Naples, Italy, 2017.

[15] J. Camacho, P. Padilla, P. Garcia-Teodoro, and J. Diaz-Verdejo, "A generalizable dynamic flow pairing method for traffic classification," *Computer Networks*, vol. 57, no. 14, pp. 2718–2732, 2013.

[16] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, and B. Xie, "Internet traffic clustering with side information," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 1021–1036, 2014.

[17] M. Shafiq, Z. Tian, A.K. Bashir, X. Du, M. Guizani, CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques, IEEE Internet Things J. 8 (5) (2020) 3242–3254.

[18] M. Shafiq, Z. Tian, A.K. Bashir, X. Du, M. Guizani, IoT malicious traffic identification using wrapper-based feature selection mechanisms, Comput. Secur. 94 (2020) 101863.

[19] E. Raff, C. Nicholas, An alternative to ncd for large sequences, lempel-ziv jaccard distance, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 1007–1015.

[20] M. Shafiq, Z. Tian, Y. Sun, X. Du, M. Guizani, Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city, Future Gener. Comput. Syst. 107 (2020) 433–442.

[21] A. Mohaisen, O. Alrawi, M. Mohaisen, Amal: High-fidelity, behavior-based automated malware analysis and classification, Comput. Secur. 52 (2015) 251–266.

[22] M. Polino, A. Scorti, F. Maggi, S. Zanero, Jackdaw: Towards automatic reverse engineering of large datasets of binaries, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2015, pp. 121–143.

[23] A. Tamersoy, K. Roundy, D.H. Chau, Guilt by association: large scale malware detection by mining file-relation graphs, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 1524–1533.

[24] L. Chen, T. Li, M. Abdulhayoglu, Y. Ye, Intelligent malware detection based on file relation graphs, in: Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), IEEE, 2015, pp. 85–92.

[25] A. Abusitta, M.Q. Li, B.C. Fung, Malware classification and composition analysis: A survey of recent developments, J. Inf. Secur. Appl. 59 (2021) 102828.

[26] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot, Sensors 17 (9) (2017) 1967.

[27] A.A. Diro, N. Chilamkurti, Distributed attack detection scheme using deep learning approach for Internet of Things, Future Gener. Comput. Syst. 82 (2018) 761–768.

[28] N. Moustafa, B. Turnbull, K.-K.R. Choo, An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things, IEEE Internet Things J. 6 (3) (2018) 4815–4830.

[29] L. Aversano, M.L. Bernardi, M. Cimitile, R. Pecori, L. Veltri, Effective anomaly detection using deep learning in IoT systems, Wirel. Commun. Mob. Comput. 2021 (2021).

[30] S.K. Sarma, Optimally configured deep convolutional neural network for attack detection in internet of things: impact of algorithm of the innovative gunner, Wirel. Pers. Commun. 118 (1) (2021) 239–260.

[31] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, in: Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on, IEEE, 2015, pp. 11–20.

[32] A.F. Agarap, Deep learning using rectified linear units (relu), 2018, arXiv preprint arXiv:1803.08375.

[33] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Icml, 2010.

[34] G.E. Dahl, J.W. Stokes, L. Deng, D. Yu, Large-scale malware classification using random projections and neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 3422–3426.

[35] W. Huang, J.W. Stokes, MtNet: a multi-task neural network for dynamic malware classification, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2016, pp. 399–418.

[36] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2016, pp. 137–149.

[37] I. Ullah, Q.H. Mahmoud, Design and development of RNN anomaly detection model for IoT networks, IEEE Access 10 (2022) 62722–62750.

[38] X. Zhou, Y. Hu, J. Wu, W. Liang, J. Ma, Q. Jin, Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial iot, IEEE Trans. Ind. Inform. (2022).

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.

[40] R. Kale, Z. Lu, K.W. Fok, V.L. Thing, A hybrid deep learning anomaly detection framework for intrusion detection, in: 2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security, IDS, IEEE, 2022, pp. 137–142.

[41] A. Abusitta, M. Bellaiche, M. Dagenais, T. Halabi, A deep learning approach for proactive multi-cloud cooperative intrusion detection system, Future Gener. Comput. Syst. 98 (2019) 308–318.

[42] A. Abusitta, O.A. Wahab, T. Halabi, Deep learning for proactive cooperative malware detection system, in: Edge Intelligence Workshop, Vol. 711, 2020, p. 7.