International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(2) 2026 pp. 460-491
https://doi.org/10.70917/ijcisim-2026-2850
Article

Enhancing Real-Time Cyber Threat Detection Using Generative Adversarial Networks (GANs)

Edward John J. P.[1], Umadevi V.[2]

[1]PG & Research Department of Computer Science, Nehru Memorial College, Bharathidasan University, Puthanampatti, Tiruchirappalli, Tamil Nadu, India.
Email: edwardjohn.jp@gmail.com
ORCID: 0009-0001-3187-7978
[2]PG & Research Department of Computer Science, Nehru Memorial College, Bharathidasan University, Puthanampatti, Tiruchirappalli, Tamil Nadu, India.
Email: yazh1999@gmail.com

Abstract: -The rapid evolution of "Internet of Things (IoT)" and 5G/6G technologies has amplified network interconnectivity, and also intensified exposure to sophisticated cyber threats. Advances in automated decision-making systems and remote data monitoring have resulted from the IoT and fifth and sixth generation (5G) technologies, which have enabled high communication rate. Some challenges arise including signature-based Network Intrusion Detection System (NIDS) struggles to adapt to zero-day exploits, risk in network traffic patterns, limited training, Inaccurate feature extraction and selection, Lack of cyber-attack detection. To tackle these issues, the research paper proposes an advanced real-time cyber threat detection framework integrating Generative Adversarial Networks (GANs) with Pigeon-Inspired Optimization and Tanh Function (SYN-GAN-PIA-TH) to enhance detection accuracy for both internal and external intrusions. The system architecture Processing (CEP) for automated preprocessing, and a ("Binary Gravitational Search Algorithm- Binary Grey Wolf Optimization") with Linear Discriminant Analysis" (BGSA-BGWO-LDA) metaheuristic ensemble for optimized "feature extraction" and "dimensionality reduction". "Long Short-Term Memory (LSTM)" models analyse temporal traffic patterns to mitigate bias and false alarms, while GSCV-SMOTE (Grid-Search Cross Validation with Synthetic Minority Oversampling Technique) addresses class imbalance in minority attack detection. Experimental results validate that synthetic data generated by GANs can effectively simulate diverse attack behaviors and achieving the high detection rate. The proposed approach establishes a robust, adaptive, and resource-efficient framework for real-time IoT network protection against emerging cyber threats. The results shows that proposed model outperforms significantly cyber threat detection using GAN, used metrics as detection rate, Number of Epoch, accuracy, Precision, F1-score and Recall.

Keywords: GAN, Cyber Threat detection, Complex Event Processing (CEP), BGSA-BGWO-LDA, GSCV-SMOTE, SYN-GAN-PIA-TH

1. INTRODUCTION

The cyber threat identification and "Real time threat intelligence" with components of significant environments of the enhancement in cyber security and integrates the real time threat intelligence and defense adaptive mechanism significantly decreases the respond to threats and the time need for detection [1]. The "Intrusion detection system" become more important tool for detection and defend the network attacks through the malicious attack with the use of security on the internet [2]. The "Cyber-attack techniques" have become more sophisticated and the attacks are increased frequently and it make more complex [3]. The IDS can analyse and monitor the total traffic transmitted to the network and the system logs are keep tracked. The IDS is an application for security that acknowledge the security breaches by the system in outsider and insider [4]. A two stage "cyber threat intelligence" using two detectors: the first detector is "Intrusion detection" and the second detector is "cyber threat intelligence". The "Intrusion detection framework" performance is based on GAN detecting accuracy and the resistance to adversarial "Evasion attacks" [5]. Some challenges are faced by the GANs are non-convergence, training instability and mode collapse. In both supervised and unsupervised learning, the GANs has been extensively employed [6].

The framework, which is based on self-attention mechanisms and GANs, that creates the adversarial malicious records in traffic included generator and the discriminator that anlayse from the "Blackbox IDS" its real time output is identified and provides some feedback from the training of generator [7]. Some issues in the deployment of GAN based systems into commonplace cybersecurity environments such as necessity for resilient scalable and the computational complexity occurs [8]. A GAN model has been added to the "Machine Learning (ML)" IDS model to ensure robustness and the GAN are used to begin a "adversarial perturbation" is considered in the network data for the threaten efficiency [9]. NIDS is a specialized tool for analysing the data packets, identifies the unauthorized or fraud activities, designed to monitor the network traffic and the main role is safeguarding networks against the extending the threat [10]. GAN have been used in recent years in anomaly detection, data generation, data reconstruction, classification, and other work [11]. The simpler methods like "Deepfool", "Carlini-Wagner (CW) attack", "Fast Gradient Sign Method (FGSM)" are used to compare the GANs where is used to generate complex adversarial and more diverse examples that are harder and overfitted on the model [12].

The network traffic patterns are analysed by the model in real time and identify the adverse actions which is linked to those patterns and using sophisticated algorithm for categorize them in new model then the pattern recognition techniques is used [13]. The "Generative adversarial network (GAN)" is an emerging technique for providing that makes helpful context for the development of DL and its used too well capture the normality that the given data its most valuable contribution has been ensured. The normality is provided under the given data and it's used to detect the abnormal instances especially using the cyber threat detection [14]. Network traffic is rapidly analysed by the NIDS and both outbound and inbound. When detecting the network threats, the NIDS aims to identify the suspicious or malicious activity [15]. The "Machine learning (ML)" based systems requires the reliable, large and valid traffic datasets to be effective and an "Intrusion Detection System" is an important defence tool for increasing the network attacks and using the sophisticated algorithm [16]. An IDS is a security tool used to analyse the network activity and the effective "Intrusion detection system" is needed for the infrastructure from different kind of the cyber-attacks [17]. The intrusion detection system is achieved its objectives by using the central server which initializes the global model and the model is well trained on the local data that is occupied by the individual nodes. The FL based "Intrusion Detection System (IDS)" which operates on the system architecture without the data exchange between the participant nodes and the central server. The characteristics of the FL- based IDS which provides and declares the exceptional choice over the AI techniques where the participate node is used to share the actual data with the centralized server [18].

To overcomes from these challenges the proposed model presents GAN, Complex Event Processing (CEP), BGSA-BGWO-LDA, GSCV-SMOTE and SYN-GAN-PIA-TH model.

A. Motivation & Objectives

The aim of this study is to implement advanced real-time the detection of cyber threat system and significantly improves the reliability and accuracy of detecting both external and internal intrusions in IoT environments. The main scope of this study was including the developing the system architecture and implementing the novel models. This study seeks to contribute to the cybersecurity domain by offering the robust solution that addresses the limits of current intrusion detection systems.

- **Boundary weakness and Partition deficit**: In existing methods using the network construction are not taken into account to increase the effectiveness of the test input space's equivalent partition and boundary value analysis.
- **Issues in preprocessing**: Existing methods have some limitations that are manual preprocessing work that manual preprocessing that will leads to the affect the detection rate and overall accuracy.
- **Inaccurate feature extraction and selection**: Previous techniques extracted a predetermined set of features from network traffic data, which might not have captured every feature that could be present in the behavior of the network. The accuracy of the recommended approaches could be improved by utilizing more sophisticated feature extraction techniques.
- **Risk in Network traffic patterns:** Current techniques tend to produce false alarms and show bias towards specific data sets. Here, the information about the patterns of network traffic as of right now is combined with the use of attack and intrusion detection tools.
- **Challenges in training:** Existing methods misclassified the detection minority attack classes that is consider as a primary issues.
- **Insufficient in attack detection**: Existing methods may not always be adequate for all attacks. Therefore, this is a need to examine other possible ways. In order to detect malicious behaviors and internal and

external intruders, the assets can be utilized in the real-time system when devising a novel intrusion detection system.

The primary objective for this research is to enhance security and optimize the transmission control unit for wireless network data transfer. The objectives in this study are as follows,

- To utilize preprocessing methods to automate and enhance preprocessing tasks, thus improving the accuracy and detection rate of cyber-attacks.
- To apply novel methods for optimizing feature selection and that novel also for detailed feature analysis to enhance the accuracy of threat detection.
- To implement effective methods to analyze and adjust for biases in network traffic patterns, thereby reducing the occurrence of false alarms and improving detection reliability.
- To use innovative technique to balance training data and improve the detection rates for minority attack classes, addressing previous misclassification problems.
- The effective novel method is combined for accurately detecting the internal and external intrusions and overall system performance is improved.

B. Research Contribution

These are illustrations of the research study's highlights:

- The “Grid-based network” is constructed to tackle the issues of boundary analysis and unequal partition.
- Use of CEP rules, the enhance preprocessing tasks are used to provide more efficiency and accuracy in cyber-attack detection.
- To utilize the novel BGSA-BGWO optimizes feature selection, while LDA analyzes detailed network features to enhance detection accuracy and avoid dimensionality issues.
- To employ LSTM networks, analyze network traffic patterns and adjust for biases, reducing false alarms and improving detection reliability.
- To utilize the GSCV-SMOTE balances the training data and improves detection rates for minority attack classes, addressing previous misclassification issues.
- To implement the SYN-GAN-PIA-TH combines synthetic data generation and advanced optimization techniques to enhance real-time detection of internal and external threats in IoT environments.

C. Paper organization

The remainder of this research as follows: Provides an explanation of Section II of a survey of previous work. In section III, states that main issue with the current methods. In section IV, defines the system model. Then in Section V presents study approach for suggested model, appropriate diagrams, mathematical representations and pseudocode. In Section VI, the suggested and current methodology is compared and experimental results are explained. The proposed model conclusion and future work is explained in Section VII.

2. LITERATURE SURVEY

In this research, they investigate The novel “semi supervised learning approach” to identify the suspicious activity by leveraging the multi traffic model characteristics. The topological information and integrating the sequence traffic, then the model achieves the topological feature from the network traffic. This model ensures the robustness and the detecting the anomalies with the encrypted traffic. The estimation modules which enhance the training set labels ratios and find the presence of unknown attacks. However, the framework has to improve the interpretability of traffic features and the “deep learning” is powerful into its end-end to learning capabilities. “Poor interpretability” can limit the application of traffic features [19]. One important feature of machine learning-operated intrusion detection systems is the capacity to transmit data securely within wireless mesh networks. If the system is subjected to an excessive number of attacks, its computational efficiency might be severely compromised. Equipped with an elliptical curve cryptography (ECC) system and a quantum neural network (QNN) built on the “whale with cuckoo search optimization (WCSO)” algorithm, the system can detect attacks and launch defenses with ease. For more precise intrusion detection, the whale optimization algorithm (WOA) is employed to select network data characteristics. Attack detection is accomplished by use of the optimized quantum network.

The WOA method, in addition to feedforward and reverse propagation algorithms, are all part of this network. Encryption is necessary for the recovery of sensitive data, and the ECC algorithm permits this operation. The data files saved on the server may be safely preserved by this procedure. In order to safeguard the paperwork via the use

of security measures, this is essential. The data owner encrypts the document using encryption technology in case sensitive information is stored on a server. Using ECC, one may find the optimal key. It is possible that the work constraint that has been stated is a challenge to the processing time with the inclusion of QNN [20]. Five distinct machine learning classifications for various attack types are developed in this work. They made use of the CSE-CIC-IDS2018 dataset, which was produced with the "Canadian Institute for Cybersecurity and the Communications Security Establishment". It's a massive collection of recently released network traffic traces that records many assaults. KDDCup'99, a dataset that has been in use for over 22 years, is the most recent noteworthy dataset that was used to create network intrusion detection algorithms. It precedes social networking, mobile computing, Web 2.0/3.0, streaming video, and the broad use of SSL. Using a restricted set of valid characteristics, they developed "decision trees, random forests, Gaussian naive Bayes, support vector classifiers", and multi-layer perceptron in this research to identify different types of network intrusions [21].

The article presents the "generative adversarial network (GAN)" that is based on a "deep learning algorithm" capable of creating the model which is implemented. The adversarial of traffic network are used to evade the detection using the "machine learning algorithm". "Intrusion detection systems (IDSs)" which is used to protect the network patterns re used to detect the network traffic. The dataset NSL-KDD dataset is used to train the model and performance is evaluated. Nevertheless, extracting relevant features from the packets and feeding them to the machine learning is difficult and classification delays occurs , it affect the performance [22]. This paper suggests a transfer-learning strategy based on SDN. Experiments later shown that our approach works better than conventional machine learning techniques. First, the framework they used scored 0.71 for identifying anomalies in unidentified assaults; second, it scored 0.98 and 0.51 for anomaly detection and attack type identification for small samples; third, it scored 1.00 and 0.91 for Class imbalance detection for anomalies and assault mechanism identification. Amongst the six models, our model ranked second in terms of efficiency and performance after training for 15,230 seconds (4 hours 13 minutes 50 seconds). Minimize sample differences across classes to address class imbalance. They will also study attack type crucial characteristics to improve multi-class classification and unknown assault detection for little amounts of data [23]. An IDS based on transformers and transfer learning was suggested for networks with unbalanced traffic in this study. Network feature representation and feature interactions in unbalanced data are both learned by IDS-INT using transformer-based transfer learning. Originally, descriptions of network interactions provide detailed information on all forms of attacks, including information about hosts, nodes in the network, attacker types, sources, and so on. In order to learn the precise feature representation, the transformer-based learning by transfer technique.

Third, the "Synthetic Minority Oversampling Technique (SMOTE)" is used to identify minority assaults and balance unusual traffic. Lastly, the CNN model looks for deep features that can be taken out of the networking flow that is balanced. In the end, several attack types are identified from the deep features by using the CNN-LSTM hybrid approach To identify different kinds of assaults on network edges, federated learning an advanced deep learning approach will be used to construct a client-server communication architecture [24].

3. PROBLEM STATEMENT

The numerous existing works and their associated responses are arranged in sequence of publication in this section. Furthermore, this study offers the research solutions for the mentioned issues.

Specific research work & Issues: Authors in [25] research paper the DL-based IoT security risk model can enable the computer to acquire from attacks on its own and accomplish continuous model training by using the real data set. In addition, the EC guarantees computer system security by accelerating data processing and model analysis, lowering necessary network traffic, and cutting down on waiting times. Additionally, EC is capable of successfully defending information security and individual privacy. The article [26] presents A hybrid novel features reduced "approach-based intelligent cyber-attack detection" system for IoT networks. Using the "correlation coefficient", "random forest" mean decrease "accuracy", and "gain", technique are used to perform feature ranking ratio in order to generate three distinct feature sets Algorithms for cyberattack detection, including "XGBoost", "K-nearest neighbor", and "random forest". The effectiveness of the two of the most "IoT-based datasets", "BoT-IoT" and NSL-KDD, are used to evaluate the suggested cyber-attack detection framework and DS2OS. According to this study [27], this research uses a Ridge Classifier as a potent model to identify anomalies in Internet of Things systems. Using safe and current network data, the suggested security system can effectively detect and anticipate cyberattacks in real time by utilizing this strategy. The defence against a variety of threats is offered by the combination of multiple security measures and robust model integration node will be the factors that define the utilization factor. Some of the problem detected in these papers are:

- Nevertheless, employing network construction techniques to enhance the effectiveness of equivalent partition and boundary value analysis of the test input space is not taken into consideration.
- The methods have some limitations that are manual preprocessing work that manual preprocessing that will leads to the affect the detection rate and overall accuracy.
- Nevertheless, a predetermined set of features were extracted from the network traffic data in this study, which might not have captured all potential features of the network behavior. The accuracy of the recommended approaches could be improved by utilizing more sophisticated feature extraction techniques.

In article [28], A "Deep neural network (DNN)" that was trained from the features of "NSL-KDD" dataset makes up the suggested system. It also includes the ML pipeline, which uses sequential components for "feature scaling" and categorical data encoding before sending "real-time data to the trained DNN model" for prediction. Additionally, a real-time feature extractor is set up in the network link between the local area network (LAN) and the gateway router. In article [29], the author proposes as a component of the security architecture, this paper suggests integrating an IDS ML approach into the 5G core architecture. This paper comparisons ML and DL procedures for intrusion detection and provides a brief overview of intrusion detection datasets. In article [30], the They suggest a novel approach based on deep learning for identifying cybersecurity flaws and breaches in cyber-physical systems. The discriminative model which is based on "deep learning" and "unsupervised learning" are contrasted in the proposed framework. In order to recognize cyber threats in IoT-driven IICs systems, this paper offers a GAN.

Some issues are focused in this research include:

- The methods exhibit some bias to some data and has a model tendency to generate false alarms. Here, the information about the patterns of network traffic as of right now is combined with the use of attack and intrusion detection tools.
- However, the proposed methods misclassified the detection minority attack classes that is consider as a primary issues.
- Moreover, these suggested methods may not always be adequate for all attacks. Therefore, this is a need to examine other possible ways. In order to detect malicious behaviors and internal and external intruders, the assets can be utilized in the real-time system when devising a novel intrusion detection system.

Research solution: To overcome from these issues, by using the grid-based network partition to overcome the abovementioned issues like equivalent partition and boundary value analysis. the manual preprocessing issues, we using the effective processing methods called Complex Event Processing (CEP) rules are accurate, and are able to perfectly detect the attacks. For feature extraction and selection using the novel method called ("Binary Gravitational Search Algorithm- Binary Grey Wolf Optimization") and Linear Discriminant Analysis (BGSA-BGWO-LDA). Here BGSA-BGWO to get an enhanced set of features to overcome the curse of efficient learning in dimensionality and for feature extraction using the LDA that will analyse the all features about the network behaviour so the proposed BGSA-BGWO-LDA has effectively analyze the possible features of the network behavior.

Traffic Pattern analysis using the "Long Short-Term Memory (LSTM)" that analysis both weight and bias that also analyze the network traffic pattern in real-time. Using the Random Forest hyperparameters are tuned with Grid-Search Cross Validation (GSCV) with Synthetic Minority Oversampling Technique (GSCV-SMOTE) that model to effectively training the better minority attack detection rates. To accurately detect the real- time system are used to detect the "internal and external intruders" and their "malicious behaviors" using novel techniques called synthetic data generated via Generative Adversarial Networks with Pigeon-Inspired Algorithm optimized with the Tanh function (SYN-GAN- PIA-TH) offers an efficient and effective solution for enhancing intrusion detection, specifically in Internet of Things environments.

4. SYSTEM MODEL

The proposed model provides a framework efficiently for "real time cyber threat detection" in IoT- enabled networks using a "hybrid deep learning and optimization approach". The architecture integrates Complex Event Processing (CEP), hybrid metaheuristic-based feature selection, temporal analysis using LSTM, data balancing through Grid searches cross validation with SMOTE (GSCV-SMOTE), and classification using a Generative Adversarial Network optimised by Pigeon inspired Optimization with Tanh activation.

A. Network and data model

The IoT environments consists of $N_d = 50IoT$ devices, two "small base stations" (SBS), one "macro base station" (MBS), and a central server. Each device generates communication packets that form diverse network traffic patterns. The generated traffic rate from node i at time t is expressed as:

$$R_i(t) = \frac{P_i(t)}{\mathbb{T}_i(t)} \quad (1)$$

Where $P_i(t)$ denotes the packet size (in bits) and $\mathbb{T}_i(t)$ is the transmission time. $R_i(t)$ is a real time transmission rate. The total aggregated traffic arriving at the base station is:

$$R_{agg}(t) = \sum_{i=1}^{N_d} R_i(t) \quad (2)$$

N_d is a total number of devices or nodes. Captured data are streamed to the central server where CEP preprocessing is performed to extract relevant temporal events. The event arrival pattern follows a Poisson distribution with rate λ_e , and the probability of observing k (number of events) events in time t is:

$$P(kT) = \frac{(\lambda_e T)^k e^{-\lambda_e T}}{k!} \quad (3)$$

B. Complex Event processing (CEP) preprocessing

CEP filters, aggregates, and correlates incoming raw event streams into meaningful event tuples. The output of the CEP engine for each observation window $\mathbb{W}$ is represented as:

$$E_w = f_{CEP}(X_{\mathbb{W}}) = \phi(F_{filter}(X_w), F_{agg}(X_w), F_{corr}(X_w)) \quad (4)$$

Where $X_{\mathbb{W}}$ represents input event data within window $\mathbb{W}$. $F_{filter}, F_{agg}, F_{corr}$ denote the filtering, aggregations, and correlation functions respectively. This process ensures that only significant and high relevance features are passed to the feature selection stage.

C. Hybrid feature selection using BGSA-BGWO-LDA

A hybrid optimization is performed by combing ("Binary Gravitational Search Algorithm (BGSA))" and ("Binary Grey Wolf optimization (BGWO))" integrated with Linear Discriminant Analysis (LDA). Each candidate solution (feature subset) is represented as a binary vector $X = [x_1, x_2, \ldots . x_n]$ Where $x_j = 1$ if feature j is selected. The fitness function combining accuracy and dimensionality reduction is given by:

$$F(X) = \alpha \times Acc(X) + \beta \times LDA_{ratio}(X) - \gamma \times \frac{|X|}{n} \quad (5)$$

Where α, β, γ are weighting coefficients, $ACC(X)$ is classification accuracy, $LDA_{ratio}(X)$ denote the discriminant capability of selected features, and $\frac{|X|}{n}$ represents the proportion of features chosen. The position of each particle is updated using BGSA and BGWO hybrid velocity dynamics to identify the optimal subset.

D. Temporal analysis Using LSTM

The Features are selected and forwarded to the "Long Short-term Memory (LSTM)" network to model sequential dependencies. The "input gate, forget gate and output gate" at time t are formulated as:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (6)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (7)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (8)$$

W_i, W_f, W_o are "weight matrices" for the input vector x_t. $The\ U_i, U_f, U_o$ are "recurrent weight matrices" for the "previous hidden state"h_{t-1} and b_i, b_f, b_o are bias vector. This architecture effectively captures time dependent behaviour within network traffic for attack identification

E. Minority Attack class Training with GSCV-SMOTE

Due to data imbalance in real world attack datasets, the "synthetic Minority oversampling Technique (SMOTE)" is used to generate "synthetic samples" for attack classes(minority). For any two minority samples x_i and x_{NN}, a synthetic instance is generated as:

$$x_{new} = x_i + \delta(x_{NN} - x_i) \quad (9)$$

Where $\delta \in [0,1]$ is a random interpolation factor. "Grid Search Cross validation (GSCV)" tunes hyperparameters such as "learning rate", "number of layers", neuron count" and maximizing F1-score during training.

F. Classification Using GAN-PIO-Tanh

The classification stage uses a Generative Adversarial Network (GAN) optimized with the pigeon- inspired Optimization (PIO) algorithm and $Tanh$ activation fr improved convergence and adversarial data generation.

PIO optimizes the generator parameters θ_G based on velocity and position updates inspired by pigeon's inspired approach:

$$V_{t+1} = V_t e^{-R_t} + rand(0,1)(P_{best} - X_t) \quad (10)$$
$$X_{t+1} = X_t + V_{t+1} \quad (14)$$

$V_{t+1}$is the updated velocity and $V_t$is the "current velocity". e^{-R_t} is a decay factor and $rand(0,1)$ is a "random number" and X_t, current position. The $Tanh$ activation enhances non-linearity and bounds generator output within [-1,1]:

$$A(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (11)$$

$A(x)$ denotes the activation function output. The final decision for classifying a traffic instance as benign or attack is given by:

$$y_{\text{pred}} = \begin{cases} 1, & \text{if } P_{\text{attack}}(x) \geq \tau \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

y_{pred} is the predicted output label and P_{attack} is the predicted probability that input x corresponds to an attack. Where τ is a detection threshold empirically selected for high recall.

G. Performance Evaluation

The proposed system performance is evaluated using "key metrics" such as "Accuracy (%)", "Precision (%)", "Recall (%)", "F1-score (%)", "Loss (%)" and "detection rate (%)". Plots are generated to analyse "the relationship between the number of epochs" and each evaluation metric, validating the learning and detection capability of the proposed hybrid model.

5. PROPOSED METHODOLOGY

The proposed methodology enhances intrusion detection in IoT environments through grid-partitioned network construction, optimized feature extraction, and real-time classification using advanced techniques like SYN-GAN-PIA-TH. These innovations ensure robust detection of both internal and external threats. The proposed model overall architecture is shown in (Fig. 1).

- Network Construction
- Real time dataset collection and Preprocessing
- Feature extraction via Network Behaviour
- Analysis of Network Traffic Pattern

- Training with Minority Attack Classes
- Real-Time Classification for Internal and External Intruder Detection

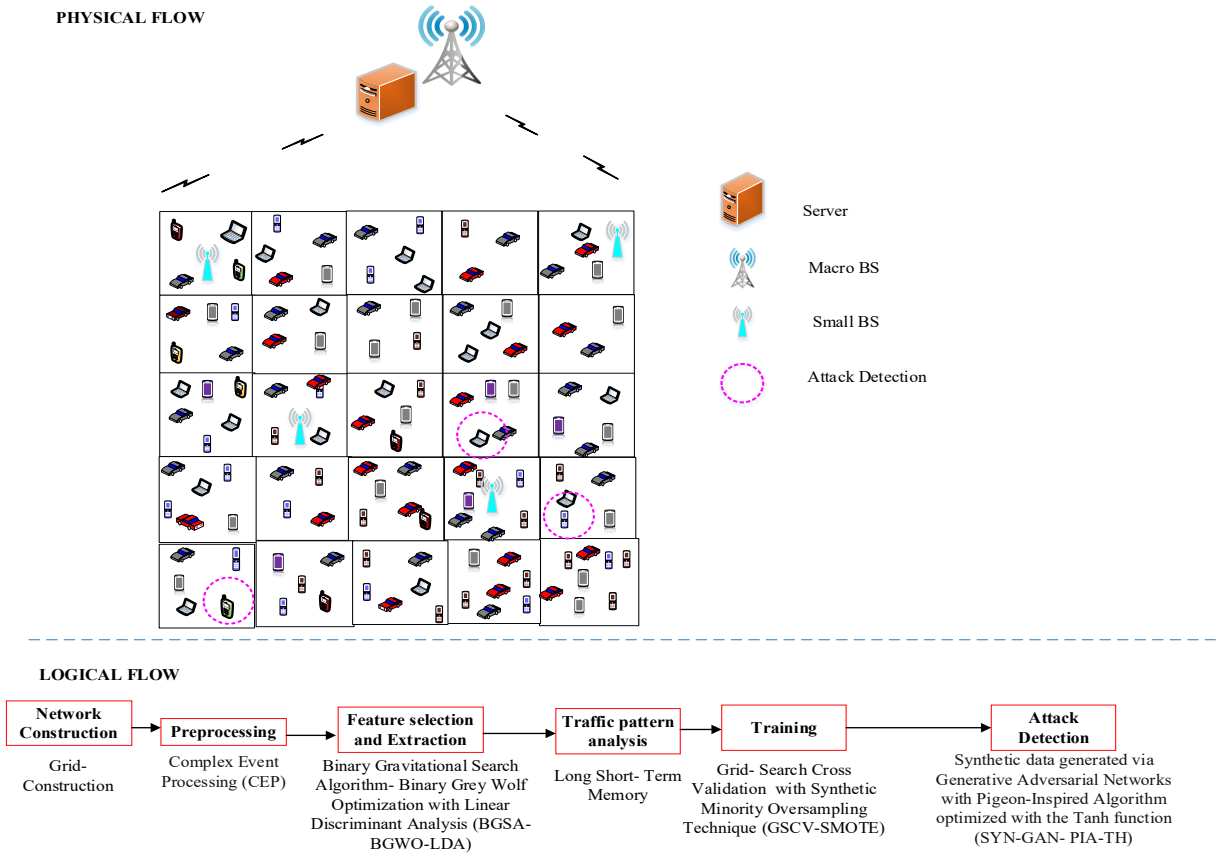

Fig. 1 Overall Proposed Architecture

A. Network Construction

Initially we construct a network is based on the **Grid- Partition based Network Partition** that are 2 dimensional the grid equally divided into Square shaped grid. Instead of computing the entire complex network topology, by using the location data of nodes and grids as the foundation of the algorithm, the overall energy consumption, boundary analysis, and can be saved. The integration of Grid-partitioned network construction with complex Event processing (CEP) offers a unified mathematical foundation for spatial temporal cyber threat analytics. Grid partitioning captures the spatial heterogeneity of network topology, while CEP rules interpret the temporal dependencies among events. The combined formulation enables localized anomaly estimation and global correlation of cyber-attack behaviours in real-time environments.

$$\mathcal{G} = \{G_{i,j} | 1 \le i \le M,\ 1 \le j \le N\} \qquad (13)$$

G represents grid-based network structure or grid partition matrix. Where $G'_{i,j}$(grid cell) is the $i^{th} \times j^{th}$ grid cell ("i "," row index" and "j", "column index"). "M, N" is a number of horizontal and vertical partitions respectively. Each grid acts as a local subnetwork responsible for monitoring its resident nodes. Node set within a Grid,

$$N_{i,j} = \{n_k | n_k \in G_{i,j}\} \qquad (14)$$

$N_{i,j}$ denotes the set of network nodes located in cell $G_{i,j}$and n_k represents an individual sensor or IoT device. The Euclidean inter node distance

$$d_{m,n} = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2} \qquad (15)$$

$d_{m,n}$ is the distance between nodes m and n. $(x_m, y_m), (x_n, y_n)$ is a cartesian coordinates of the respective nodes. The metric quantifies connectivity potential inside a grid.

$$E_{G_{i,j}} = \sum_{n\epsilon N_{(i,j)}}(E_{\mathbb{T}x,k} + E_{rx,k} + E_{idle,k}) \quad (16)$$

$E_{\mathbb{T}x,k}, E_{rx,t}, E_{idle,k}$ denote transmission, reception and idle mode energy of node k. $E_{G_{i,j}}$ is a total energy consumption. The cumulative energy guides routing and node action policies. In Boundary connectivity indicator

$$B_{i,j} = \begin{Bmatrix} 1, & d_m \leq R_{th} \\ 0, & otherwise \end{Bmatrix} \quad (17)$$

R_{th} is a communication radius threshold. $B_{i,j}$ (binary connectivity indicator) denotes 1 implies that neighboring nodes maintain direct connectivity, ensuring grid integrity. d_m is the distance between two communicating entities.0 indicates inactive connectivity that is node are beyond the communication range. The Grid mapping function is represented as,

$$\prod(n_k) = (i,j) if\ n_k \in G_{i,j} \quad (18)$$

$\prod(n_k)$ maps each node to its respective grid coordinates, enabling spatial indexing for event assignment. The normalized Grid weight:

$$W_{i,j} = \frac{|N_{i,j}|}{\sum_{p=1}^{M}\sum_{q=1}^{N}|N_{p,q}|} \quad (19)$$

$W_{i,j}$ is the normalized weight or probability coefficient. The local event density is formulated as,

$$\curlywedge_{i,j}(t) = \frac{1}{|N_{i,j}|}\sum_{n_k \in N_{i,j}} e_{m_k}(t) \quad (20)$$

$\curlywedge_{i,j}(t)$ is a temporal rate of security events within grid $G_{i,j}$. The $e_{m_k}(t)$ is the number of detected abnormal packets by node n_k at time t. This expresses spatially localized attack intensity. The complex Event processing (CEP) Modelling interprets continuous event streams by temporal windowing, aggregation and rule evaluation. Event stream representation can be generated as,

$$S = \{E_t - (T_t, a_t, x_t)\}_{t=1} \quad (21)$$

Srepresents system state set or sequence of state representations. Each event E_t(estimated energy level) comprises timestamp T_t(temporal component) activity type a_t(action parameter). $t = 1$ denotes the initial time step. The attribute vector $x_t = [x_{t1}, x_{t2}, \ldots . x_{td}]$. Window function per grid:

$$W_{i,j}(t) = \{E_r \in S | t - \Delta\mathbb{W}_l \leq r \leq t, E_r \in G_{i,j}\} \quad (22)$$

E_r is an energy observation and measures network state at a specific time instance r. $\Delta\mathbb{W}_l$ is a window length and $W_{i,j}(t)$ aggregates events in the last Δt seconds within grid $G_{i,j}$. And r is the temporal index variable. The event level aggregation:

$$f_{i,j}(t) = \frac{1}{|W_{i,j}(t)|} \quad \sum_{E_t \in W_{i,j}(t)} \varphi(E_t) \quad (23)$$

$\varphi(E_t)$ represents feature extractor yielding metrics such as packet rate or entropy. $f_{i,j}(t)$ is a mean feature vector describing the grid's instantaneous behaviour. The anomaly deviation score is analysed as:

$$A_{i,j}(t) = \left|\left|f_{i,j}(t) - \mu_{i,j}\right|\right|_2 \quad (24)$$

$\mu_{i,j}$ is a running mean of historical features. $A_{i,j}(t)$ quantifies deviation from normal traffic in Euclidean space. The CEP correlation pattern rule as,

$$P_r: (E_n \rightarrow E_b)[\Delta t] \wedge \psi(x_a, x_b) \quad (25)$$

P_r detects a casual sequence where event E_a is followed by E_b within interval Δt satisfying predicate ψ; The x_a and x_b represents spatial coordinates of two interacting entities, grid cells or nodes. The Rule-matching indicator,

$$M_r(t) = \begin{Bmatrix} 1, & if\ P_r\ satisfied\ in\ W_{i,j}(t) \\ 0, & otherwise \end{Bmatrix} \quad (26)$$

$M_r(t)$ (binary satisfaction indicator) signals whether r is activated during the current window. The weighted correlation score is generated as:

$$C_{i,j}(t) = \sum_{r=1}^{R} w_r M_r(t) \quad (27)$$

$C_{i,j}(t)$ accumulates all satisfied rules. W_r is a confidence weight or historical reliability of rule r; R is a total number of CEP rules. The integrated Grid-CEP analytics, spatially derived metrics from grid construction and temporally derived features from CEP are fused to assess local and global threat probabilities. The Grid level threat index is formulated as,

$$T_{i,j}(t) = \alpha A_{i,j}(t) + \beta c_{i,j}(t) \quad (28)$$

α, β is a weighting coefficient balancing anomaly magnitude and rule correlation. $T_{i,j}(t)$ are a local threat index forms a composite indicator of link within the grid $G_{i,j}$. The global Threat probability:

$$P_{threat}(t) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \sigma(T_{i,j}(t)) \quad (29)$$

$\sigma(.)$ is a sigmoid activation mapping scores to probability range [0,1]. $P_{threat}(t)$ expresses the network -wide instantaneous threat likelihood. The online statistical update as,

$$\mu_{i,j}(t+1) = (1-\eta)\mu_{i,j}(t) + \eta f_{i,j}(t) \quad (30)$$

$$\sigma_{i,j}^2(t) = (1-\eta)\sigma_{i,j}^2(t) + \eta \left(f_{i,j}(t) - \mu_{i,j}(t)\right)^2$$

η is an exponential smoothing constant $(0 \leq \eta \leq 1)$. These recursive relations ensure adaptive learning of normal behaviour for each grid. The CEP driven has be represented as,

$$\delta_{i,j}(t) = \begin{Bmatrix} 1, & T_{i,j}(t) \leq \theta_{th} \\ 0, & otherwise \end{Bmatrix} \quad (31)$$

θ_{th} denotes the decision threshold derived from validation statistics. $\delta_{i,j}(t) = 1$, initiates a real time grid $G_{i,j}$. The overall detection efficiency:

$$\varepsilon_{det} = \frac{\sum_t \sum_{i=1}^{M} \sum_{j=1}^{N} \delta_{i,j}(t)}{\sum_t \sum_{i=1}^{M} \sum_{j=1}^{N} 1} \quad (32)$$

ε_{det} is the ratio of successfully detected anomalies to total evaluated events; higher values indicate superior detection accuracy. Pseudocode for Grid-partition based Network is presented in Algorithm 1.

Algorithm 1: Grid- Partition based Network framework

1. **Initialize** total nodes, grid size and radius
2. Create empty grid structure
3. **For each** node i:
4. Assign random position(X, Y)
5. Place node in its grid cell
6. **For each** pair of nodes in same grid:
7. Compute distance$d(i,j)$
8. **If** $d(i,j) \leq radius \rightarrow connect\ nodes$
9. **Else→** no connection
10. Compute node energy = $T_x + R_x + Idle$
11. Sum all node energies for total energy
12. Check boundary nodes for connectivity
13. Count abnormal nodes for connectivity

14. Compute local event density
15. Apply CEP rule to detect anomalies
16. Calculate local threat score per grid
17. Combine all grid for global threat score
18. **If** $threat\ score \geq threshold \rightarrow Alert$
19. End

B. Real time dataset collection and Preprocessing

After network construct and Realtime- data collection the collected data were preprocessing using the "**Complex Event Processing (CEP)"** allows to be automatically applied. The former uses artificial intelligence (AI) to identify "attack patterns in real time", while the uses of "Principal Component Analysis (PCA)" algorithm to characterize events and identify anomalies. Then results demonstrate that CEP rules obtained using approach significantly outperform the "standard CEP", which the rules are not generated by the proposal and is easily and independently analyse a subject matter expert. This combination enables the achievement of "efficient CEP rules" at "computational level". Incoming packet flow observations as an event stream, apply time-aware windowing, perform feature extraction and normalization, handle missing/ erroneous values, and produce labelled feature vectors for CEP rule engines and machine learning. Fig. 2 represents the Dataset collection and pre-processing. The following equations give a combined, reproducible mathematical description of the CEP pipeline. Stream model and primitives:

$$S = \{e_i\}_{i=1}^{\infty}\ ,\ e_i = (t_i, a_i, \pi_i) \tag{33}$$

The continuous event stream S as a sequence of events e_i. Each event is a tuple containing a timestamp t_i, an activity/type label a_i and a payload or attributes vector π_i

$$t_i \epsilon \mathbb{T}, \mathbb{T} \subseteq \mathcal{R} \geq 0 \tag{34}$$

Timestamps t_i are really valued non negative time instants (monotonic ingest order, subject to clock skew); $\mathbb{T}$ denotes the time domain and $\mathcal{R}$ is a set of non-negative real numbers.

$$\pi_i = \left[x_{i,1}, x_{i,2} \dots \dots, x_i d_x\right]^T \tag{35}$$

$[x_{i,1}, x_{i,2} \dots \dots, x_i d_x]$ is an individual feature component of i^{th} entity index and T is the transpose operator. The d_x is the "dimension of the feature vector". The raw attribute vector π_i has d_x fields,

$$\mathbb{W}_t^{\Delta} = \{e_i \epsilon S | t - \Delta < t_i \leq t\} \tag{36}$$

Define a tumbling or sliding time window $\mathbb{W}_t^{\Delta}$ of size Δ terminating at current time t. It contains events arriving in the last Δ seconds.

$$\mathcal{G}_t^{\Delta} = group(W_t^{\Delta}, k(\pi)) \tag{37}$$

$\mathcal{G}_t^{\Delta}$ denotes resulting group set and $k(\pi)$ is a feature-based similarity. Group events in W_t^{Δ} by a keying function $k(\pi)$ to produce groups $\mathcal{G}_t^{\Delta}$ for per- entity aggregation.

$$\mathcal{G}_j = \{e \epsilon W_t^{\Delta} | k(\pi_e) = k_j\} \tag{38}$$

$\mathcal{G}_j$ denotes the j^{th}group formed from the spatio temporal window W_t^{Δ}. Each group $\mathcal{G}_j$ contains events sharing the same key k_j.

$$agg_f(\mathcal{G}_j) = \varphi_f(\{\pi_e : e \in \mathcal{G}_j\}) \tag{39}$$

An aggregation operator agg_f computes feature f for group $\mathcal{G}_j$ via extractor φ_f(Feature aggregation function)

$$Z_{t,j} = \left[agg_1(\mathcal{G}_j), agg_2(\mathcal{G}_j), \dots \dots agg_d(\mathcal{G}_j)\right]^T \tag{40}$$

$Z_{t,j}$ is a final aggregated feature vector and d is the "dimension of the group level feature vector". Construct the raw feature vector $Z_{t,j} \epsilon \mathbb{R}_d$ for group $\mathcal{G}_j$ at time t by concatenating d_z aggregated statistics.

$$IAT_{ek} = t_k - t_{k-1} \quad (41)$$

$\forall_{ek}\epsilon\mathcal{G}_j$ ordered by t .Inter arrival times (IAT) for events in a group are computed to capture temporal burstiness; t_k is the "timestamp of the current event" e_k and t_{k-1} is the "timestamp of the previous event". These enter aggregators such as mean, max of IAT.

$$\Phi_{seq}(G_j) = LSTM(\{\pi_e\}_{e\epsilon G_j}\epsilon\mathbb{R}_d \quad (42)$$

$\mathbb{R}_d$ is a real value dimensional space and $\{\pi_e\}_{e\epsilon G_j}$ represents sequence of feature vectors for all entities. $\Phi_{seq}(G_j)$ is a sequential feature vector representing temporal dependencies within grid G_j. Optionally compute a sequence embedding via an LSTM encoder Φ_{seq} that maps ordered event attributes to d_t is a dimensional temporal feature vector. Normalization, scaling and encoding.

$$Z_{t,j} = norm(z_{t,j}; \mu, \sigma) = \frac{Z_{t,j}-\mu}{\sigma+\varepsilon} \quad (43)$$

Standardize aggregated features using running mean μ and standard deviation σ; ε is a small constant to prevent division by zero.

$$\mu_t = \alpha_\mu \mu^{t-1} + (1-\alpha_\mu)Z_t \quad (44)$$

The $\mu_t$is the smoothed feature vector and α_μ is the smoothing factor for the previous smoothed state. Maintain online estimates of feature mean via exponential moving average with smoothing α_μ. Z_t is the batch mean at time t.

$$\sigma_t = \alpha_\sigma \sigma^{t-1} + (1-\alpha_\sigma) std\ (Z_t) \quad (45)$$

σ_t updated standard deviation at time step t and σ^{t-1} is the standard. α_σ is the smoothing factor. $std\ (Z_t)$ is the instantaneous standard deviation of the current observation set Z_t. Similarly update running standard deviation with smoothing α_σ.

$$\mathbb{M}(c)\epsilon\{0,1\}^c \quad c = con(\pi) \quad (46)$$

$\mathbb{M}(c)$ is a message bit sequence of length c and $\epsilon\{0,1\}^c$ is a binary domain of dimension d. Categorical attributes are encoded via embedding lookups for categorical feature c extracted from π. $con(\pi)$ represents content length function of pseudonym π.

$$Z_{t,j}^{imp} = impute(Z_{t,j}) \quad (47)$$

Apply imputation to missing entries of normalized feature vector $Z_{t,j}^{imp}$ using rule-based, mean or model-based imputation strategies. imp is an imputation function or operator. j is a sensor index.

$$isOutlier(x) = 1(|x-\mu_2| >\!\!\succ_0 \sigma_x) \quad (48)$$

Flag outliers via a threshold $\succ_0$ on standardized features $1(|x-\mu_2| >\!\!\succ_0 \sigma_x)$ is the indicator function. σ_x is the standard deviation of x and μ_2 is the mean value.

$$Z_{t,j}^{clean} = clip(Z_{t,j}^{imp}, l, u\ ) \quad (49)$$

$Z_{t,j}^{clean}$ is the final pre-processed for time t and variable j. Feature Clip the feature values to limits l, u derived and mitigate spurious extremes.l denotes lower bound and u is the upper bound.

$$pattern\ P: (e_a \rightarrow e_b) \wedge \Phi(\pi_a, \pi_b) \quad (50)$$

A CEP rule P specifies a pattern where event e_a is followed by e_b within temporal constraint and satisfying predicate Φ on attributes; such rules detect multi-event. $\wedge$ is a logical AND operator and $\Phi(\pi_a, \pi_b)$ is a logical or statistical condition that relates the attributes of the related events e_a

And e_b.

$$matches\ p(t) = \{(e_i, e_j)\epsilon\mathbb{W}_t^\Delta | t_j - t_i \leq \Delta t,\ \varphi(\pi_i, \pi_j) = 1\} \quad (51)$$

$matches\ p(t)$, refers to pair of events (e_i, e_j) and Evaluate rule p over window $\mathbb{W}_t^{\Delta}$ to produce matched event tuples that satisfy temporal and attribute predicates. $(e_i, e_j) \epsilon \mathbb{W}_t^{\Delta}$ represents the window of events around time t and $t_j - t_i \leq \Delta t$ is a temporal proximity.

$$C_p(t) = \sum_{(e_i,e_j)\in matches\ p(t)} w(\pi_i, \pi_j) \qquad (52)$$

$C_p(t)$ represents cumulative score associated with pattern p Aggregate match scores for pattern p at time t. Using weight function $W(\pi_i, \pi_j)$ producing the scalar correlation.

$$X_{t,j} = \left[Z_{t,j}^{clean}, \varphi_{seq}(\mathcal{N}_j); \{C_{pk}(t)\}_{k=1}^{m}\right] \in \Re^d \qquad (53)$$

The final pre-processed feature vector $X_{t,j}$ by concatenating cleaned aggregated features, optional LSTM sequence embedding, and the vector of m CEP pattern correlation scores. $\Re^d$ denotes that the final vector $X_{t,j}$ is d-dimensional and $\{C_{pk}(t)\}_{k=1}^{m}$ is the set of cumulative pattern scores for m patterns at time t. The labelling, buffering and output. $\varphi_{seq}(\mathcal{N}_j)$ represents sequential feature representation of graph G_j.

$$l_{t,j} = l(x_{t,j}, t_{anom}, \mathcal{K}) \qquad (54)$$

Assign provisional label $l_{t,j}$ using a labelling function $l(.)$ that may combine thresholding on anomaly score t_{anom}, known blacklists $\mathcal{K}$, or online signature matches; labels may be refined offline.

$$\mathcal{B} \leftarrow \mathcal{B} \cup \{(x_{t,j}, l_{t,j}, t)\} \qquad (55)$$

Append the pre-processed and labelled record to an output buffer $\mathcal{B}$ for downstream tasks. $\cup$ is the union operator. Buffering supports backpressure and batch writes.

$$latency(t) = t - t_{ingest}(e_i) \qquad throughput = \frac{\mathfrak{T}}{\Delta_t} \qquad (56)$$

$latency(t)$ denotes the processing delay and$t_{ingest}(e_i)$ is a timestamp when event e_i was ingested into the system. $\mathfrak{T}$ is a number of events processed and $\frac{\mathfrak{T}}{\Delta_t}$ is the average rate of processing events. Measure per event end to end latency and system throughput over internal Δ_t as operational constraints; CEP must maintain bounded latency and sufficient throughput for real time detection.

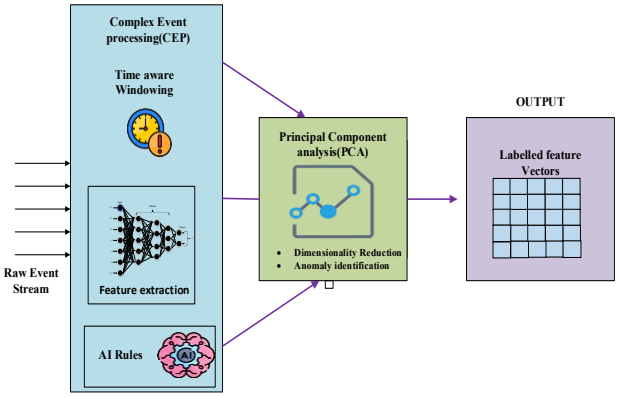

Fig. 2 Dataset collection and Preprocessing

C. *Feature extraction via Network Behaviour*

The next step is to preprocess the feature selection using the metaheuristic-based ensemble feature selection framework, which is created using the Binary Grey Wolf Optimization and "Binary Gravitational Search Algorithm" to obtain an optimal dimensionality use for "effective learning". And that features are extracted by using the Liner Discriminant Analysis that reflect modern network behaviour are utilised to conduct. In Fig. 3, the feature extraction via Network behaviour is shown. Here using the novel called ("**Binary Gravitational Search Algorithm- Binary Grey Wolf Optimization") with Linear Discriminant Analysis (BGSA-BGWO-LDA)** has effectively analyze the possible features of the network behavior. The proposed methodology introduces a hybrid metaheuristic model

integrating the binary gravitational search algorithm (BGWO) with linear Discriminant Analysis (LDA) for optimal feature extraction and classification enhancement in real time cyber threat detection systems. The model aims to maximize class separability and detection accuracy while minimizing redundant or irrelevant network features.

Initially, each feature vector is represented as a binary selection:

$$m = [m_1, m_2, \dots\dots, m_d]\ m_j \in \{0,1\} \tag{57}$$

m is the binary feature selection vector and m_j is the binary variable for the j^{th} feature. $\mathbb{D}_f$ represents total number of features in the dataset. $m_j \in \{0,1\}$ is the feature selected set and $m_j = 1$ indicates that the j^{th} feature is selected. The dataset after selection is defined as:

$$X_m = X.diag(m) \tag{58}$$

X represents the matrix that contains the original data. X_m contains the featured selected matrix and $diag(m)$ is the diagonal matrix created from vector m. The within-class scatter matrix for LDA is computed as:

$$S_w = \sum_{c=1}^{C}(x_i - \mu_c)(x_i - \mu_c)^T \tag{59}$$

S_w is within class scatter matrix. The x_i, "data point" in class c and μ_c is the "mean vector of class c". $x_i - \mu_c$ is used to captures how far each sample is from the class centre. T is the transpose of the column vector. The class scatter matrix is given by:

$$S_B = \sum_{c=1}^{C} n_c(\mu_c - \mu)(\mu_c - \mu)^T \tag{60}$$

S_B represents "between class scatter matrix" and μ is the "overall mean vector" of all samples from all classes . n_c denotes the "number of samples" in class c. μ_c is the "mean vector of class" c. The discriminant objective for maximizing class separability is:

$$J_{LDA} = tr((S_w + I)^{-1}S_B) \tag{61}$$

J_{LDA} is used to measure the ratio between class scatter to within class scatter and tr, coverts the matrix into scalar score that measures overall class separation. $((S_w + I)^{-1}S_B)$ represents a normalized separation of classes considering within class variability. Each agent in BGSA-BGWO-LDA represents binary subset. The fitness function is evaluated as a combination of classification accuracy and LDA discriminant ratio:

$$F(m) = \ell.Acc(m) + m.\frac{J_{LDA}}{1+J_{LDA}} - \curlywedge \frac{\sum_j m_j}{\mathbb{D}_f} \tag{62}$$

F is the objective function in feature selection and $Acc(m)$ is the classification accuracy obtained using the selected features m. Where ℓ and $m$are weighting parameters and $\curlywedge$ penalizes the "number of selected features" and $\mathbb{D}_f$ is the "number of features in the dataset".$\sum_j m_j$ denotes "the number of selected features". In the BGSA phase, the mass of each agent k is defined as:

$$M_k(t) = \frac{fit_k(t) - worst(t)}{best(t) - worst(t) + \delta} \tag{63}$$

$M_k(t)$ is the normalized fitness or performance score of the k^{th} candidate feature at iteration i'. $fit_k(i')$ represents original fitness value of the k^{th} candidate at iteration i' $best(t)$ is the "best fitness value" among all candidates at iteration t. $Worst(t)$ is the "worst fitness value" among all candidates at iteration t . δ is a "small positive number added to avoid division" by zero if $best(t) = worst(t)$.

$$g(t) = g_o \exp(-\gamma \frac{i'}{T_{max}}) \tag{64}$$

$g(t)$ is parameter value g at iteration i'. g_0, initial value of g at iteration $i' = 0$. γ is the "decay rate" or scaling factor.T_{max} is the maximum time or "total number of iterations".

$$F_k(t) = \sum_{i=1, i\neq k}^{s} rand_{ik}.g(t)\frac{m_k(t)m_i(t)}{||P_i(t) - p_k(t)||_2 + \eta} \tag{65}$$

$F_k(t)$ is the force, influence, or effect acting on the k^{th} agent at iteration t and sum of overall agents i expect k. $rand_{ik}$ is used to prevent getting stuck in local optima. $m_k(t)m_i(t)$ is a product of fitness based or selection weight of agent k and i at iteration t. $P_i(t) - p_k(t)$ represents the Euclidean distance between positions of agent of i and k at iteration t.The "velocity" and "position" are updated as:

$$V_k(t+1) = w_v V_k(t) + a_k(t) \quad (66)$$

$V_k(t+1)$ is the "velocity" of the k^{th} agent at the next "iteration" $t+1$. W_v is the inertia weight or scaling factor for the velocity.$V_k(t)$ is the current "velocity" of the k^{th} agent at "iteration" t. $a_k(t)$ denotes the acceleration on agent k at iteration t.

$$P_{kj}(t+1) = 1, if \ \frac{1}{1+\exp(V_{k,j\,(t+1))}} > r_j \quad (67)$$

$P_{kj}(t+1)$ is the feature selection bit of the k^{th} agent for the j^{th} feature at the iteration $t+1$. $V_{k,j\,(t+1))}$is the velocity of agent k for feature j at iteration $t+1$. r_j is the "random number" in [0,1] for the j_{th} feature. In the BGWO phase, the leadership hierarchy is maintained by alpha, beta and delta wolves and their positions are updated as

$$D_{t,k} = |C_t . W_{l_l} - W_k| \quad (68)$$

$D_{t,k}$ is the distance between two values at iteration t for agent or feature k. C_t is a scaling or influence factor at iteration t. W_f, "weight of feature" and W_k, weight vector or parameter k. Then the reference point l.

$$X_{l,k=}w_l - A_l . D_{t,k} \quad (69)$$

$X_{l,k}$ is "updated position" or value k^{th} agent in the l^{th} dimension. A_l , coefficient or scaling factor, often controlling the step size or influence of the distance term.

Where,

$$A_t = 2ar_1 - a\,, \quad C_t = 2r_2, \quad a = 2\left(1 - \frac{\alpha_m}{T_{max}}\right)^K \quad (58)$$

The A_t is a coefficient that controls the step size and α_m is the coefficient controlling the magnitude.a is a decreasing control parameter over iterations. r_1, r_2 is a random number.

And the binary update is defined as:

$$W_{k,j}(t+1) = \begin{cases} 1, & \text{if } \sigma\left(\frac{X_{a,k,j}+X_{s,k,j}+X_{a,k,j}}{3}\right) > r_j \\ 0, & \text{otherwise} \end{cases} \quad (70)$$

$W_{k,j}(t+1)$ is the updated value of the weight and $X_{a,k,j} + X_{s,k,j} + X_{a,k,j}$ is the input feature score or contributions for the j^{th} feature of entity k. r_j is the random threshold for feature j. The proposed hybridization integrates both BGSA and BGWO updates using an adaptive mixing coefficient:

$$\mu(t) = \mu_o \left(1 - \frac{t}{T_{max}}\right)^k \quad (71)$$

$\mu(t)$ represents parameter value at iteration t. $\mu(0)$ is the initial parameter value at $t = 0$. k is the exponent controlling the rate of decay and T_{max} is the "maximum number of iterations". The unified position updates the hybrid model is computed as:

$$s_k(t+1) = binarize\ (\mu(t) p_k^{BGSA}(t+1) + \big(1 - \mu(t)\big) w_k^{BGWO}(t+1) \quad (72)$$

$s_k(t+1)$ denotes updated binary position of the k^{th} iteration at t. $binarize$ is the function that converts a continuous value into binary 0 or 1. $\mu(t)$, balances the contributions of BGSA and BGWO over iteration. $p_k^{BGSA}(t+1)$ denotes the candidate solution by using "BGSA algorithm". $w_k^{BGWO}(t+1)$ represents "candidate solution" by the BGWO algorithm. Elitism ensures the best solution is retained by:

$$p_k(t+1) = s_k(t+1), if\ rand > \rho \quad (73)$$

$p_k(t+1)$ represents the updated position of k^{th} . $s_k(t+1)$ is a computed value at iteration t. ρ is the threshold probability. The global best solution is selected as:

$$m* = \arg\max F(p_k(t)) \quad (74)$$

$p_k(t)$ denotes the k^{th} "candidate solution" at iteration t. $m*$ is the "index of the solution" that gives the maximum fitness and $arg\ max$, it returns the index at which the function reaches its maximum.

Finally, the optimal projection matrix is generated using:

$$W_{final} = W_{LDA}(m*) \tag{75}$$

W_{final} is the final weight vector or matrix selected for classification or feature projection and W_{LDA} is the linear discriminant analysis.

And the reduced feature vector for classification is:

$$z = W_{final}^{T}(x \odot m^{*}) \tag{76}$$

$\odot$ denotes element wise multiplication and z is the projected feature vector. x denotes an "input vector". W_{final}^{T} represents transpose of the matrix. A proposed BGSA-BGWO-LDA methodology efficiently balances the exploration and exploitation, improves discriminant capability and optimizes feature selection for enhanced cyber threat detection accuracy.

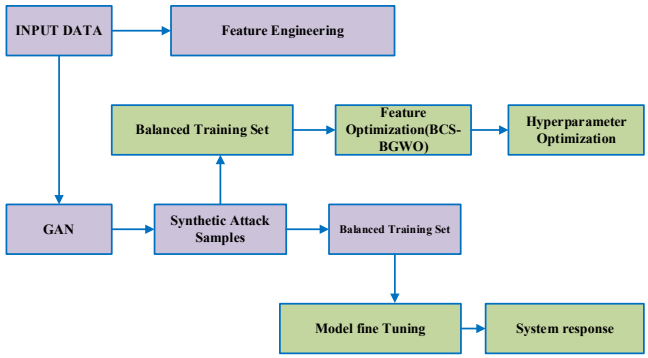

Fig. 3 Feature Extraction

D. Analysis of Network Traffic Pattern

After completing the feature selection and extraction next, analyze the network traffic pattern using to "improve the robustness" of the technique to detecting the IoT cyberattacks, **Long Short- Term Memory (LSTM)** deep models were combined with their outputs to analyze network traffic patterns. After completing the feature extraction and selection process using BGSA-BGWO-LDA, the next step is to analyse the network traffic pattern using "Long short-Term Memory (LSTM)" model. The "LSTM architecture" is analysed for sequential traffic data, "capturing temporal correlations" and "long-term dependencies" to accurately identify dynamic cyber threat behaviours. The mathematical representation of the proposed LSTM based network traffic analysis is formulated as follows:

$$x_t = [f_1, f_2, \ldots . f_d]; \quad x_t \epsilon \mathbb{R}^d \tag{77}$$

Where x_t represents the input feature vector extracted from BGSA-BGWO-LDA and d denotes the total number of optimized features. $[f_1, f_2, \ldots f_d]$ is the list of individual features and $\mathbb{R}^d$ is the d dimensional real vector space. The model processes a sequence of such feature vectors to learn temporal dependencies and sequential relationship in the network traffic.

$$f_t = \sigma(W_f[h_{t-1,}x_t] + b_f) \tag{78}$$

The "forget gate" f_t controls which information from the "previous hidden state" should be discarded.

Here, $\sigma(W_f[h_{t-1,}x_t] + b_f)$ represents "sigmoid activation function", W_f is the "weight matrix" and b_f is the "bias vector". The "forget gate" helps eliminate irrelevant or outdated information from the model's memory.

$$i_t = \sigma(W_i[h_{t-1,}x_t] + b_i) \tag{79}$$

The input gate(i_t) determines how much of the current input information should be added to the memory cell. The term W_i represents "weight matrix", and b_i denotes the "bias for the gate". This step enables LSTM to incorporate recent traffic characteristics that might indicate potential attack behaviour.

$$c_t = \tanh(W_c[h_{t+1}, x_t] + b_c) \quad (80)$$

The "candidate cell state" c_t generates new potential information to be stored in the cell. h_{t+1}, x_t denotes concatenation of the "previous hidden state".x_t is the input feature vector at "current time t".The $tanh()$ activation ensures the values remain within a stable range of [-1,1], which aids numerical stability and helps model nonlinear dependencies in network activity.

$$c_t = f_t \odot c_{t-1} + i_t \odot c_t \quad (81)$$

"cell state c_t" serves as the long-term memory of the model. It is updated and combining the previously extracted information ($f_t \odot c_{t-1}$) and new information ($i_t \odot c_t$). This mechanism enables the model to retain important temporal relationships across time steps, which is essential for detecting long duration or evolving cyber-attacks.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (82)$$

The output gate O_t regulates which portion of the current cell state should be revealed to the output. b_o is the "bias vector" of output gate and then x_t ,the input feature vector at current time t. This gating mechanisms ensures that only relevant, filtered information contributes to the next hidden state for accurate temporal prediction.

$$h_t = o_t \odot \tanh(c_t) \quad (83)$$

The hidden state h_t captures the current temporal representation of the network's behaviour at time step t. It reflects both the new input and the retained historical memory. C_t is the cell state at time t. This output is passed to the next time step and to the final classification layer.

$$y_t = softmax(W_y h_t + b_y) \quad (84)$$

The $softmax$ is an activation function generates the probability distribution for different traffic classes. W_y is the "weight matrix for the output layer" and b_y is the "bias vector for the output layer". h_t is the "hidden state vector" from the LSTM at time step t. It converts the hidden state into interpretable class probabilities use for attack prediction and classification.

$$L = \sum_{t=1}^{T} \log(y_t) \quad (85)$$

The "loss function (L)" measures the difference between the "predicted class probabilities" and labels y_t. The cross-entropy loss is minimized during training to improve prediction accuracy and reduce classification error.

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} \quad (86)$$

The weight update rule uses gradient descent to iteratively adjust model parameters. $\frac{\partial L}{\partial W}$ is the "gradient of the loss function L with respect to weight W".Here, η represents the learning rate the determines the step size during optimization.

$$\rho_{t,t+k} = \frac{Cov(h_t, h_{t+k})}{\sqrt{Var(h_t) Var(h_{t+k})}} \quad (87)$$

The temporal correlation coefficient $\rho_{t,t+k}$ quantifies the relationship between hidden states at different time intervals. $Cov(h_t, h_{t+k})$ is the covariance between the "hidden state at time" and hidden state at time $t+k$. $Var(h_t)$ is the variance of the hidden state at time t. $Var(h_{t+k})$ is the variance of the hidden state at time $t+k$. A high correlation value indicates the presence of long-term dependencies or recurring traffic behaviours within the observed network sequence.

$$z_t = \Phi(W_z) \quad (88)$$

z_t is the output vector. The feature temporal fusion combines the optimized features from BGSA-BGWO-LDA with the temporal output h_t. The transformation function φ(Non-linear transformation) produces a compact

joint representation that reflects both spatial (feature based) and temporal (sequence based) patterns of the traffic. The W_z is the weight matrix.

$$D_{input} = [z_t, y_t] \tag{89}$$

This equation defines the input (D_{input}) to the GAN discriminator, which receives both the fused representation z_t and the predicted class output y_t. This allows the discriminator to difference between "real and synthetic temporal traffic patterns "generated by GAN.

$$L_D = \mathbb{E}_z[\log D(x)] - \mathbb{E}[\log(1 - D(G(z, h_t)))] \tag{90}$$

The discriminator loss L_D("Loss function of the discriminator") and $\mathbb{E}_z$ denotes expectation over real time data x. $\log D(x)$evaluates how effectively the discriminator can distinguish between genuine and generated data. $g(z, h_t)$ is the "generator function" that maps noise. First term corresponds to real samples, while the second term penalizes incorrectly accepted synthetic samples. Inclusion of h_t, helps maintain temporal coherence during adversarial learning.

$$L_g = \mathbb{E}_z[\log D\left(g(z, h_t)\right)] \tag{91}$$

The generator loss L_g ensures that the generator learns to produce synthetic data sequences that closely resemble real traffic patterns. This enhances data diversity and supports balanced training for minority attack classes.

$$\alpha_t = \frac{\exp(h_t W_a h_r)}{\sum_{k=1}^{T} \exp(h_k W_o h_r)} \tag{92}$$

The attention coefficient α_t,assigns importance weights to each time step. h_t represents "hidden state vector" at time step t and W_a is the weight matrix for computing the alignment score of h_t. h_r is the reference or context vector. h_k is the "hidden state vector" at time t and W_o is the weight matrix used to transform the hidden state before computing attention. Higher values of α_t indicate time intervals with more significant behavioural deviations, such as sudden bursts or unusual packet flows, where the crucial for detecting anomalies.

$$c_r = \sum_{t=1}^{T} \alpha_t h_t \tag{93}$$

The context vector c_r aggregates the attention-weighted hidden states across all time steps. It represents condensed summary of the most influential temporal features within entire observation window.

$$y_{final} = softmax(W_c c_T + b_c) \tag{94}$$

The final classification output y_{final} provides the predicted category of the network sequence, distinguishing between normal and various attack types. W_c is the "weight matrix" for the final classification layer and C_T is the "context vector" or final hidden representation at the output space.b_c is the "bias vector" for the final classification layer. The main decision output of the LSTM-based traffic analysis.

$$Conf_{attack} = \frac{e^{attack}}{\sum_j e^{ij}} \tag{95}$$

$Conf_{attak}$ is the confidence score or predicted probability for the specific attack class. Where, e_{attack} represents exponential of the logit corresponding to the attack class.$\sum_j e^{i\,j}$ is the sum of exponentials of logits for all classes. The detection confidence score measures how certain the model is about its prediction for a specific attack class. A higher confidence value indicates a stronger detection likelihood for that class.

$$\Delta W = \eta(y_t - y_t^1)h_t \tag{96}$$

The weight correlation rule adjusts the model parameters based on the prediction error$(y_t - y_t^1)$. This refinement step ensures that the LSTM model continues to improve its detection capability over successive epochs. Pseudocode for Long short-term memory is presented in Algorithm 2.

Algorithm 2: Long short-term Memory

1. Input optimized feature set F_{opt} from BGSA-BGWO-LDA

2. **Initialize** LSTM parameters: Weights (W_f, W_i, W_c, W_O) and biases (b_f, b_i, b_c, b_o)
3. Set learning rate η, $epochs\ E$, and sequence length T.
4. **For each** training sequences $s \epsilon F_{opt}$ do
5. **Initialize** hidden state $h0 \leftarrow 0$ and cell state $c0 \leftarrow 0$
6. **For each** time step $t = 1$ to T do
7. $x_t \rightarrow input\ feature\ vector\ at\ time$
8. $f_t \rightarrow \sigma\big(W_f[h(t-1), x_t] + b_f\big)$
9. Input gate as $it \rightarrow \sigma(W_i[h(t-1), x_t] + b_i$
10. Candidate cell state $C_t \rightarrow \tanh(W_c[h(t-1), x_t] + \quad bi\ )$
11. Cell update c_t
12. $o_t \rightarrow \sigma(w_0[h(t-1), x_t] + b_o)$ is the output gate.
13. h_t is a hidden state update
14. **End for**
15. $y_t \rightarrow Softmax\big(w_y . h_t + b_y\big)$
16. Compute loss $L \leftarrow Crossentropy(y_t, Y_t)$
17. Update weights: $W \leftarrow W - \eta \nabla L(W)$
18. Compute correlation ρ_t between hidden states.
19. Apply attention weight $at = softmax(W_a \tanh(h_t))$
20. **Output** predicted label $y = argmax\ (softmax(W_c))$

E. Training with Minority Attack Classes

After analysing the network traffic pattern, train the model to more effectively detect minority attacks by using **Grid-Search Cross Validation with Synthetic Minority Oversampling Technique (GSCV-SMOTE),** where the Random Forest hyperparameters are tuned. In Fig. 4, the training with Minority attack classes is shown. To create an initial GS model that achieves and trains the models efficiently, RF hyperparameters are adjusted using grid-search cross-validation. The dataset definition states that,

$$\mathbb{D} = \{(x_d, y_i)\}_{i=1}^{N}, \quad x_d \epsilon \mathbb{R}^d,\ y_i \epsilon C = \{c_1, \ldots . c_m\} \quad (97)$$

The full dataset $\{(x_i, y_i)\}_{i=1}^{N}$ contains N samples with d dimensional features and labels in m classes. $x_i \epsilon \mathbb{R}^d$, where x_d is a dimensional real valued vector and $y_i \epsilon c_l = \{c_1, \ldots . c_m\}$ is the class label and c is the set of all possible class and there is m class in total.

$$\mathbb{D}_s = \{(x_i, y_i) \epsilon D : y_i = c\}, \qquad |D_c| = n_c \qquad (98)$$

The $\mathbb{D}_s$ represents the subset of the dataset n_c is the "number of samples" in c and Isolate class specific subsets. For each class c. denote its size n_c. $(x_i, y_i) \epsilon \mathbb{D}$ is the pair from the original dataset $\mathbb{D}_o$ and $y_i = c$ is a key pair whose label $is\ c$. Minority classes have small n_c. The imbalance ratio (per class)

$$IR_c = \frac{\max_c c \epsilon n_d}{n_c} \qquad (99)$$

Where IR_c is an imbalance ration that compares the largest size to class c. Large IR_c is a severe imbalance. The $\max_c c \in n_d$ denotes the maximum operation of overall classes. The n_c is the "number of samples" in class c. The target oversampling factor (SMOTE)

$$s_c = \max(0, [\beta.(n_{max} - n_c)]) \quad (100)$$

Where, the minority class c, compute number of synthetic samples s_c (Score, weight synthetic sample count for class c. To add $n_{max} = \max n_c$; $\beta\epsilon[0,1]$ controls how much balancing to perform $\beta = 1 \Longrightarrow$full balance). n_{max}, the difference between the largest classes and class c. n_c is the largest class size. The KNN set for each minority sample. The smote synthetic sample generation (vector form)

$$x_{syn} = x + \lambda(x_{nm} - x) = \quad \lambda \sim \mathcal{U}(0,1), x_{nm}\epsilon N_k(x) \quad (101)$$

x_{syn}, synthetic sample is generated along a line segment between x and one of its KNN neighbors; $\searrow$ is uniform. $x_{nm}\epsilon N_k(x)$is a neighbor of x in the datatset. $\lambda(x_{nm} - x)$ is the vector differentiating between x_{nm} and x. Number of synthetic samples per minority instance

$$s_c(x) = \left[\frac{s_c}{n_c}\right] \quad (102)$$

The distribute class level synthetic count s_C, roughly evenly across existing minority instances $x \in D_c$. The resulting balanced dataset after SMOTE

$$\mathbb{D}' = \mathbb{D} \cup \{(x_{syn}, c_l)\} \quad (103)$$

The new dataset $\mathbb{D}'$ includes original samples plus all generated synthetic minority samples. x_{syn} is a synthetic data sample. $\mathbb{D}_o$ represent the original dataset before sampling. c_l denotes the class label and $\cup$ is a union operator. The k-fold cross validation split on $\mathbb{D}'$

$$\mathbb{D}' = \bigsqcup \mathbb{D}'_j, \quad \mathbb{D}'_{train_j} = \mathbb{D}^l \mathbb{D}'_j, \quad \mathbb{D}'_{val_j} = \mathbb{D}'_j \quad (104)$$

Split $\mathbb{D}'$ into k disjoint folds for cross-validation. For fold j to train on D'_{train_j} (training dataset) and validate on $\mathbb{D}'_{val\,j}$. $\sqcup\, \mathbb{D}'_j$ is the disjoint union of subsets. The hyperparameter grid shows that,

$$\mathbb{U} = \prod_p = \{\theta^1, \ldots \ldots \theta^H\} \quad (105)$$

$\mathbb{U}$ is a unified parameter set and $\prod$ is a cartesian or parameter product. θ^H represents the parameter vector of the H^{th} model. Construct a cartesian, Then the model training map

$$f_{\theta(g)} = A(\mathbb{D}'_{train_j}; \theta^g) \quad (106)$$

The $f_{\theta(H)}$ is the local model function and $A(\mathbb{D}'_{train_j}; \theta^H)$ is the learning algorithm. $\mathbb{D}'_{train_j}$ is the local training dataset. The per fold validation loss (or error)

$$L_j(\theta^H) = \frac{1}{|D'_{val}|} \sum_{(x,y)\epsilon D'_{val}} l(y, f_\theta(x)) \quad (107)$$

The $L_j(\theta^H)$ is the average loss computed on the validation dataset and $(y, f_\theta(x))$ is a "loss function" which is used to measure discrepancy between true label and predicted output. j represents the node and g is the model index. The $(x, y) \in \mathbb{D}'_{val}$ is the validation data sample. Compute average loss L_j on the validation fold using a chosen loss l. The cross-validation risk (mean validation loss)

$$R_{CV}(\theta^H) = \frac{2P_{j,c}(\theta^H)R_{j,c}(\theta^H)}{P_{j,c}(\theta(H)) + R_{j,c}(\theta^H)} \quad (108)$$

$R_{CV}(\theta^H)$ denotes the "harmonic mean of precision" and "recall" for the H^{th} model on class c. $P_{j,c}(\theta^H)$ is the fraction of correctly predicted samples over total predicted samples for class c in validation set j and $R_{j,c}(\theta^H)$ is the fraction of correctly predicted samples over total actual samples for class c in validation set j. The compute per class $F1$ on validation fold j for class c. c denotes the class index.P is a precision and R is a recall of the classes. Then the weighted minority focused $F1$ across folds

$$F1_{min}(\theta^H) = \frac{1}{K}\sum_{j=1}^{K} \frac{1}{|\mathcal{C}_{\min}|} \sum_{c\epsilon\mathcal{C}_{min}} F1_{j,c}(\theta^H) \quad (109)$$

$F1_{min}(\theta^H)$ is the average F1 score of rare classes and K is a number of validation partitions. $\mathcal{C}_{min}$ denotes subset of classes with minimum frequency in the dataset. The average $F1$ over validation folds, but computed only across the set $\mathcal{C}_{\min}$of minority classes. $F1_{j,c}$ is the F1 score for class c in partition j and θ^H is the model parameters for node H. The composite objective for selection

$$J(\theta^g) = R_{CV}(\theta^g) - \alpha\, F1_{min}(\theta^g) \qquad (110)$$

$J(\theta^H)$ is the objective function and $R_{cv}(\theta^H)$ is the average F1-score across all classes and validation partition for model g. $F1_{min}(\theta^H)$ is an average F1 score for the least frequent classes C_{min}in overall partitions. α is the trade-off coefficient. The combined criterion that penalizes validation loss while minority class $F1; \alpha > 0$ balances the two terms. The grid search selection:

$$\theta^* = \arg\min J(\theta^g) \qquad (111)$$

θ^* are the optimal model parameters and $J(\theta^H)$ is the objective function. The choose of hyperparameters θ^* that minimize the composite objective J. The final model retraining on full balanced set

$$f^* = A(\mathbb{D}';\ \theta^*) \qquad (112)$$

f^* represents the final trained model and $A(\mathbb{D}';\ \theta^*)$ is a training algorithm. It retrains the classifier on the entire balanced dataset $\mathbb{D}'$using the selected hyperparameters θ^*. The predicted class probabilities and decision rule

$$p(c|x) = f_c^*(x) \qquad y = \arg\max p(c|x) \qquad (113)$$

$p(c|x)$ represents "probability that input x "belongs to class c, as predicted by the trained model and$f_c^*(x)$ is the "output of the final trained model" f^* corresponding to class c(class index).where, x is an input sample and y is a predicted class label. Final classifier outputs class probabilities p is a predicted label y is the argmax. The performance of the minority macro-F1

$$Macro\ F1_{min} = \frac{1}{|\mathcal{C}_{\min}|} \sum_{c\epsilon\mathcal{C}_{min}} \frac{2P_cR_c}{P_c+R_c} \qquad (114)$$

$Macro\ F1_{min}$ represents the Macro F1 score of rare classes and P_c is the precision for class c. R_c is a recall for class c.Final reported metric for minority classes- the macro averaged $F1$ across minority classes used to demonstrate the improvement.

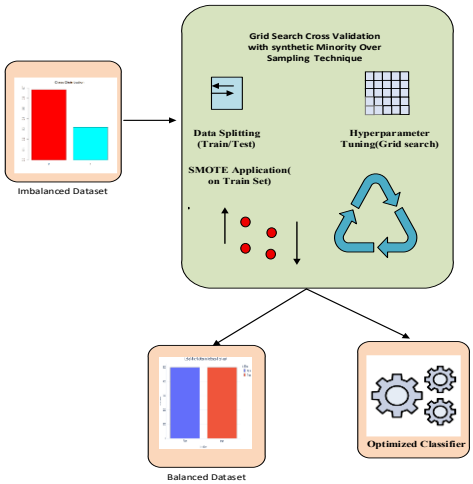

Fig. 4 Training with Minority attack classes

F. Real-Time Classification for Internal and External Intruder Detection

Our research filled a vacuum in the literature by demonstrating the feasibility of using only synthetic data generated by GANs to detect the attacks. Our approach has produced encouraging results that are on par with those from real-time datasets, casting doubt on the traditional use of real-world data. Especially in Internet of Things environments, the Pigeon-Inspired Algorithm optimized with the Tanh function provides a practical and efficient way to improve intrusion detection system performance. To accurately detect the "real- time system" for detecting the

"internal and external intruders" using the novel techniques called synthetic data generated via **Generative Adversarial Networks with Pigeon-Inspired Algorithm optimized with the Tanh function (SYN-GAN- PIA-TH)** offers an efficient and effective solution for enhancing intrusion detection, specifically in Internet of Things environments. The model hybrid model integrates the strengths of Generative Adversarial Networks (GAN), the "pigeon inspired Algorithm (PIA)" optimized via the hyperbolic tangent function(tanh) and data balancing techniques using the "synthetic Minority oversampling Technique (SMOTE)" under grid search Cross validation (GSCV).

$$\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{N} \tag{115}$$

Where $x \in \mathbb{R}^d$ represents a d "dimensional feature vector extracted from network traffic", and $y_i \epsilon c_l = \{1,2, \ldots . c\}$ denotes the corresponding attack class label. $\mathbb{D}$ is a dataset and x_i, y_i is a pair of input features x_i and label $y_i . i$ denotes the "sample index" and N represents "total number of samples". Due to the inherent class imbalance in intrusion datasets, minority attack samples are enhanced using both GAN-based synthesis and SMOTE based interpolation. A synthetic GAN(SYN-GAN) model is employed to generate realistic network attack patterns. The generator g_θ produces synthetic samples x from random latent noise $z \sim P_z(Z)$:

$$x = g_\theta(z), \tag{116}$$

x represents synthetic sample and g_θ is a generator function. z denotes random noise vector. Where θ denotes the learnable "parameters of generator network".

$$\mathbb{D}_\varphi(x) = \Pr(x \text{ is real } \varphi) \tag{117}$$

The $\mathbb{D}_\varphi(x)$ represents discriminator output and φ is the "discriminator parameters". The $\Pr(x \text{ is real} \mid \phi)$ is a scalar value between 0 and 1 indicating the likelihood that x is real. The adversarial min-max between G_θ and $\mathbb{D}$ is formulated as:

$$Min\ Max\ L_{adv}(\theta, \varphi) = \mathbb{E}_z[\log D_\varphi(x)] + \mathbb{E}_z[\log(1 - D(G_\theta\ (z)))] \tag{118}$$

$L_{\text{adv}}(\theta, \phi)$ is the objective function for generator and discriminator and θ is the general parameters and φ is the discriminator parameters. The equation compels the generator to synthesize realistic intrusion patterns such that the discriminator cannot differentiate synthetic samples from real ones. $\mathbb{D}_\varphi(x)$ is the discriminator output and $g_\theta(z)$ denotes the generator output. $\mathbb{E}_z$ is the expectation over noise. And z is the noise vector. In the case of conditional synthesis, where attack types are labelled, the generator can be extended to accept the class condition c.

$$x = g_\theta(z, c)\ , \qquad \mathbb{D}_\varphi(x, c) \tag{119}$$

D_φ is the discriminator function and c is a class label. Allowing the generation of targeted minority attack patterns. To preserve the sequential nature of network traffic, temporal dependencies are encoded through a long short-term memory (LSTM) embedding function $\varphi_{LSTM}(.)$.A temporal regularization loss ensures that synthetic and real sequences have similar feature dynamics:

$$L_{temp} = \mathbb{E}_{s,s}\left[||\varphi_{LSTM}(s) - \varphi_{LSTM}(s)||_2^2\right] \tag{120}$$

Here, s and s_1 denote real and synthetic traffic sequences, respectively. L_{temp} is used to measure the discrepancy between LSTM embeddings of related sequences and $\mathbb{E}_{s,s}$ is an average over pair. $\varphi_{LSTM}(s)$ is the representation of sequence s extracted by the LSTM network. To address severe class imbalance, the SMOTE algorithm creates synthetic samples by interpolating between minority class instances, Let the imbalance ratio for a class c be:

$$\rho_c = \frac{\max_k |\mathbb{D}_k|}{|\mathbb{D}_c|} \tag{121}$$

Where $|\mathbb{D}_k|$ denotes "number of samples" in class c and ρ_c , ratio indicating how many times the largest class exceeds class c in size. The "number of synthetic samples" required for class c is:

$$N_c^{syn} = [(\rho_c - 1)|\mathbb{D}_c|\alpha] \tag{122}$$

N_c^{syn} is the "total number of synthetic samples" generate for class c using GAN. Where $0 < \propto \leq 1$ as the balancing coefficient. Each new sample is created by interpolating between a minority instance x_i and one of its k-nearest neighbors x_{nn}.

$$x^{SMOTE} = x_i + r(x_{nn} - x_i), \tag{123}$$

x^{SMOTE} denotes new synthetic feature vector generated for the classes and x_i is the original samples. The x_{nn} is the feature vector from the same class that is closest to x_i. $r \sim \mathcal{U}(0,1)$ is a random interpolation factor. The final balanced dataset combining real, GAN-generated, and SMOTE synthetic samples.

$$\mathbb{D}^* = \mathbb{D} \cup S_{GAN} \cup S_{SMOTE} \tag{124}$$

$\mathbb{D}_o$ is the original dataset and $\mathbb{D}^*$ denotes the augmented dataset. S_{GAN} are the synthetic samples produced by the generator network to augment classes. S_{SMOTE} represents synthetic samples generated via SMOTE to balances classes. This balanced dataset is subsequently used for classifier training and hyperparameter tuning. The GSCV method systematically searches for optimal hyperparameters across a finite grid $G = \{g_1, g_2, \ldots . g_M\}$ is formulated as,

$$g^* = \arg\min_{g \epsilon \mathcal{G}} \frac{1}{K} \sum_{l=1}^{K} L_{val}(\mathcal{V}_g; V_l), \tag{125}$$

The g^* is the optimal generator and $g \in \mathcal{G}$ is a candidate generator. Where k is the number of cross-validation folds, $\mathcal{V}_g$ represents the classifier trained under configuration g, and V_l denotes the validation subset in fold l. $\arg\min$ is a minimization operator. The validation loss emphasizes minority class performance through a composite cost:

$$\mathcal{L}_{val} = W_1(1 - Accuracy) + W_2(1 - F1_{minor}) \tag{126}$$

Where $F1_{minor}$ is the mean F1-score for minority attack classes, and W_1, W_2 are importance weights. The pigeon Inspired Algorithm emulates the homing behaviour of pigeons, which navigate using map-based orientation. Let the j^{th} pigeons' position at iteration t be $p_j(t)$ and its velocity be $v_j(t)$. The PIA position update rule is formulated as:

$$P_j(t+1) = p_j(t) + v_j(t) \tag{127}$$

$P_j(t+1)$ is the "new position of particle" j at iteration $t+1$ and $p_j(t)$ is the position j at iteration t in the parameter. The $v_j(t)$ is the vector determining from $p_j(t) to\ P_j(t+1)$.The velocity consists of a component,

$$v_j^h(t) = \gamma_1(t)\big[g(t) - p_j(t)\big], \tag{128}$$

Where $v_j^h(t)$ is the velocity of particle j in dimension h at iteration t and $\gamma_1(t)$ is a learning factor. $g(t)$ is the global position and $p_j(t)$ is the current position at iteration $t. h$ is a dimension index.

$$v_j^l(t) = \gamma_2(t)[r_1 \odot \big(p_a(t) - p_b(t)\big)] \tag{129}$$

Where $g(t)$ denotes the current global position, p_a, p_b are two random pigeons, r_1 is a random vector with entries in (0,1) and $\odot$ denotes element wise multiplication. $\gamma_2(t)$ is a time dependent learning factor and $v_j^l(t)$ is the local component. The coefficient $\gamma_1(t)$ and $\gamma_2(t)$ are time-decaying parameters exploration and exploitation. To prevent divergence during optimization, $tanh$ scaling is applied,

$$v_j(t) = \tanh(\beta(t)\ [\ v_j^h(t) + v_j^l(t)] \tag{130}$$

Where $\beta(t) = \beta_o e$ dynamically reduces the search amplitude. The non linearity of $\tanh(.)$ bounds the velocity updates to (-1,1), thereby ensuring stability in generator parameter optimization. The best candidate parameter generator is updated as:

$$g(t+1) = \arg\max_j F_j(t+1) \tag{131}$$

Where $F_j(t)$ denotes "fitness function", typically the inverse of validation error. The generator total cost function combines adversarial, temporal and PIA regularization terms

capability over successive epochs. Pseudocode for Generative Adversarial Networks with Pigeon-Inspired Algorithm optimized with the Tanh function (SYN-GAN- PIA-TH) is presented in Algorithm 3.

Algorithm 3: Generative Adversarial Networks with Pigeon-Inspired Algorithm optimized with the Tanh function (SYN-GAN- PIA-TH)

1. Input: optimized feature set $X = \{X_1, X_2, \ldots \ldots \ldots X_n\}$ labels y
2. **Initialize** generator $g(z;\ \theta_g)$ and discriminator $\mathbb{D}(x)$
3. Generate synthetic samples
4. Combine dataset: $\mathbb{D}_{train} = X \cup x$
5. Apply Pigeon-Inspired optimization (PIO) to update θ_g, θ_D
6. Update velocity:$V_i(t+1)$
7. Update position:$X_i(t+1)$
8. Apply $Tanh$ activation
9. Feed real and synthetic samples into discriminator D.
10. Compute discriminator output: $\mathbb{D}(x) = \sigma(W_x + b)$
11. **For each** incoming network packet:
12. Compute real or fake probability for each sample
13. $\boldsymbol{If}$ probability ≥ threshold → classify as attack
14. **Else** → classify as "Normal"
15. Update generator and discriminator based on loss
16. Re-train with minority attack classes using GSCV-SMOTE
17. Compute accuracy, precision, recall and F1 score
18. $\boldsymbol{If}$ intrusion→ trigger alert and log event accessed.
19. Output classified results with confidence score
20. **End**

6. EXPERIMENTAL RESULTS

A. Simulation setup

The section proposes cyber threat detection using GAN as well as it is set up for simulation. To simulate the proposed model, the Python 3.11.7 is used. In Table 1, represents the System specification.

Table 1

System specifications

Hardware specifications	Hard disk	512 GB
	RAM	8GB
Software specifications	Simulation tools	Python 3.11.7
	OS	ubuntu 22.04

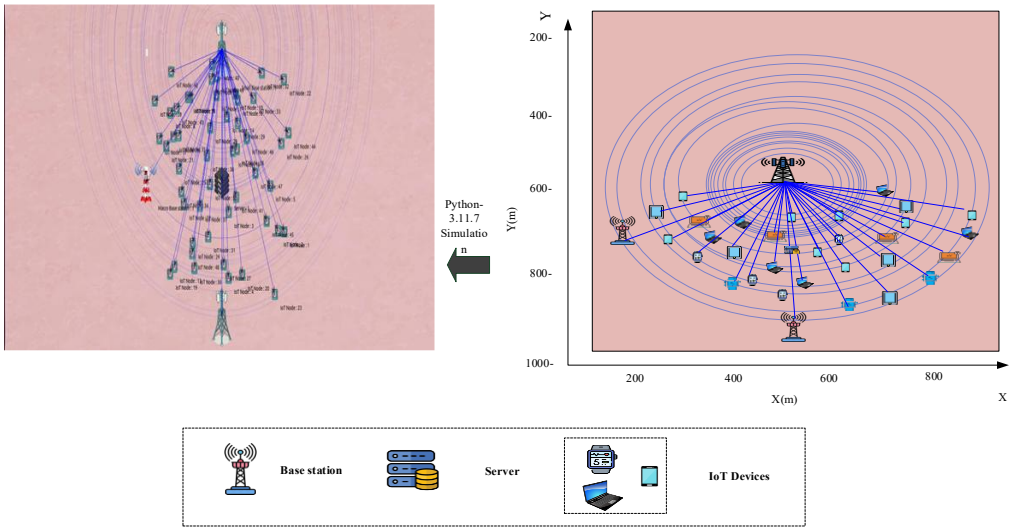

Fig. 5 NetAnim visualization process

In Fig. 5, The NetAim visualization process is shown. The loaded network topology is simulated by clicking the run button and the process is completed successfully.

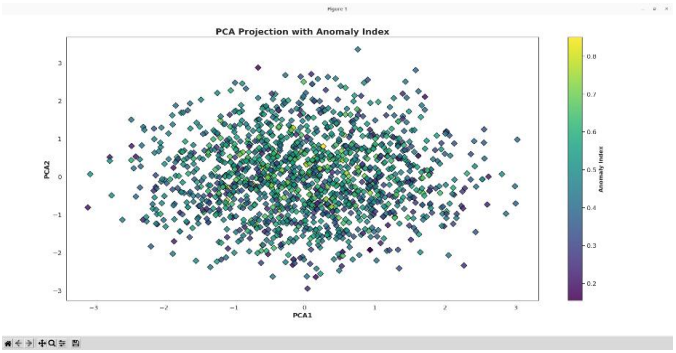

Fig. 6 PCA projection with anomaly index

In Fig. 6, PCA projection with anomaly index is shown. Principal Component Analysis (PCA) provides high-dimensional data into a lower-dimensional data, and anomaly index is derived from the "reconstruction error**".**

B. Comparative analysis

The proposed method is evaluated by comparing the existing methods in following domains: “Number of Epochs vs. Detection Rate (%)”, “Number of Epochs vs. Loss (%), Number of Epochs vs. Accuracy (%)”, “Number of Epochs vs. Precision (%)”, “Number of Epochs vs. F1- measure (%)”, “Number of Epochs vs. Recall (%)”.Compared to existing method such as CADF(Cyber-attack detection framework) [2] and DL-GAN(Deep learning Based generative adversarial model) [19] model, the proposed model performed good and accurate.

a. Number of epochs Vs detection rate (%)

An epoch represents one complete pass of the training data through the model. The detection rate measures how accurately the model identifies correct instances, such as cyber threat or anomalies. As the epochs increases, the model learns better patterns from the data, improving detection rate. Overall, a higher detection rate with optimal epochs indicates effective model learning. In Fig. 7, “Number of epochs” Vs detection rate (%) is presented.

Table 2

Number of epochs Vs detection rate

X-axis (Number of epochs)	Y-axis (detection rate (%))		
	Proposed	CADF	DL-GAN

20	85	72	62
40	88	74	64
60	91	76	66
80	94	78	68
100	97	80	70

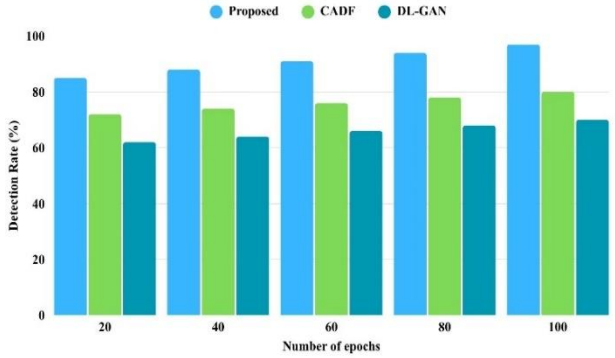

Fig. 7 Number of epochs Vs detection rate

In table 2, the "Number of epochs" and "Detection rate (%)" is shown. Initially at 20 epochs, the proposed model reaches 85% of detection rate which is better and higher than the CADF (72%) and DL-GAN (62%) respectively. Finally at 100 epochs, the proposed model archives 97%, accuracy of detection rate is higher than the CADF (80%) and DL-GAN (70%) model. The model achieves reliable threat detection and highest detection rate.

b. Number of epochs Vs LOSS (%)

The "Number of epochs" Vs Loss represents the error reduces during the training. The loss (%) denotes that the how far the methods deviate from the original data. When the number of epochs increases the model decreases the error through the weight matrices. The lower or decrease loss percentage represents the better model performance. In Table 3, the "number of epochs" Vs Loss is presented.

Table 3

Number of epochs Vs LOSS (%)

X-axis (Number of epoch)	Y-axis (Loss (%))		
	Proposed	CADF	DL-GAN
20	90	80	70
40	92	82	72
60	94	84	74
80	96	86	76
100	98	88	78

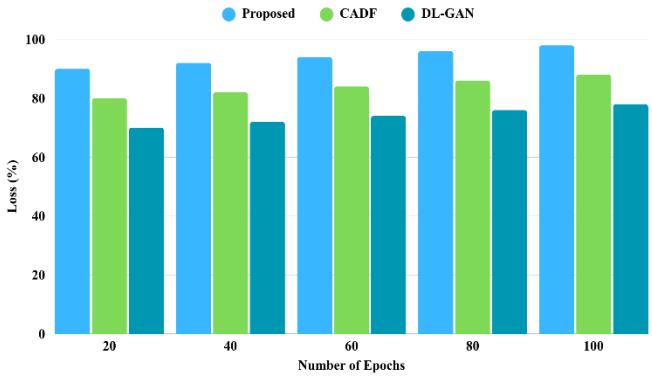

Fig. 8 Number of epochs Vs LOSS (%)

The X-axis denotes “number of epochs” and the Y-axis represents “loss (%)”. Initially At 20 epochs, the proposed method reaches the loss 90% reflects the initial prediction errors than the CADF (80%) and DL-GAN (70%). Finally, at 100 epochs, the proposed model achieves 98% loss, whereas CADF (88%) and DL-GAN (78%). The proposed method achieves higher loss growth when compared to the existing such as CADF and DL-GAN. In Fig. 8 Number of epochs Vs LOSS (%) is represented.

c. Number of epochs Vs accuracy (%)

The “Number of Epochs” Vs Accuracy shows that model’s prediction performance evolves during training. Accuracy measures “proportion of correctly predicted outputs” to total prediction. With each epoch, the model updates its parameters, leading to improved accuracy. A high accuracy value reflects effective learning and better generalization. In Fig. 9 Number of epochs Vs accuracy is shown.

Table 4

Number of epochs Vs accuracy (%)

X-axis (Number of epochs)	Y-axis (Accuracy (%))		
	Proposed	CADF	DL-GAN
20	85	70	60
40	90	72	62
60	92	74	64
80	95	76	66
100	97	78	68

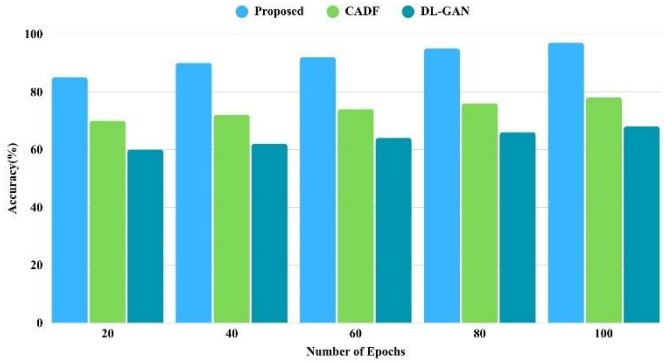

Fig. 9 Number of epochs Vs accuracy (%)

In table 4 “Number of epochs” at x-axis and model accuracy at y-axis. Initially, at 20 epochs, the proposed method shows 85% accuracy better than the existing method such as CADF (70%) and DL-GAN (60%). Finally, at 100 epochs, the proposed method reaches 97% of higher accuracy than the CADF (78%) and DL-GAN (68%). The proposed model achieves the highest and better accuracy compared to the existing method.

d. Number of epochs Vs Precision (%)

The “Number of epochs” Vs “precision” are used to identify the prediction in the model. When the number of epochs increases, the method differentiates the pattern more efficiently and is used to improve the precision. The high precision is involved the learning saturates. In Fig. 10 Number of epochs Vs precision is shown.

Table 5

Number of epochs Vs Precision (%)

X-axis (Number of epochs)	Y-axis (precision (%))		
	Proposed	CADF	DL-GAN
20	90	80	70
40	92	82	72
60	95	84	74

80	97	86	76
100	99	88	78

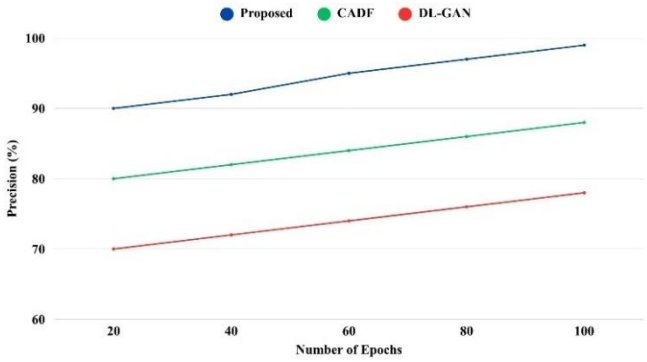

Fig. 10 Number of epochs Vs precision (%)

In table 5, X-axis represents the "number of epochs", is a training the Y-axis shows precision values. Initially, at 20 epochs, the proposed method shows 90% is higher than the CADF (80%) and DL-GAN (70%) shows that the better and accurate classification. Finally, at 100 epochs, the proposed model achieves 99% which is used to highlight the effective of detection of information than the CADF (88%) and DL-GAN (78%). The proposed model achieves the highest precision.

e. **Number of epochs Vs F1-measure (%)**

The "Number of epochs" vs "F1" measure shows that balance between the recall and precision. When the F1 is raising, the "number of epochs" is increases and then the model is identifying the true positive and reduces the false negatives. The F1 measure is "harmonic mean of precision and recall" which is used to reflect the model. In table 6, the "Number of epochs" Vs F1-measure is shown.

Table 6

Number of epoch Vs F1-measure (%)

X-axis (Number of epoch)	Y-axis (F1-measure (%))		
	Proposed	CADF	DL-GAN
20	82	70	60
40	84	72	62
60	86	74	64
60	88	76	66
100	90	78	68

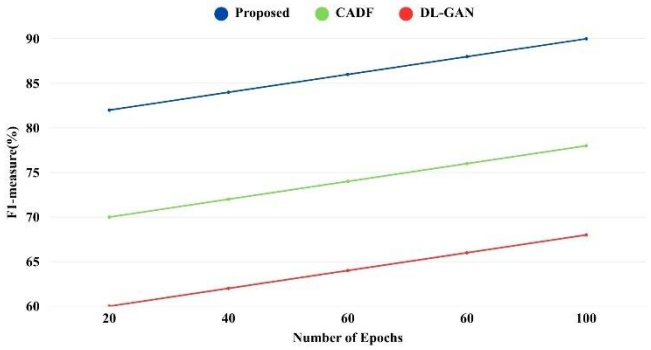

Fig. 11 Number of epochs Vs F1-measure (%)

In Fig. 11 Number of epochs Vs F1-measure (%) is represented in the graph**.** The X-axis denotes "number of epochs", for training while the Y-axis shows F1-score, which balances recall and precision. Initially 20 epochs, the proposed model shows 82% is better than the CADF (70%) and DL-GAN (60%). Finally at 100 epochs, the proposed model achieves 90%, whereas CADF (78%) and DL-GAN (68%). The proposed model achieves highly detection efficiency and F1 measure across all epochs provides better and balanced performance.

f. Number of epoch Vs Recall (%)

The "Number of epochs" Vs recall graph shows that the identification of the model in all information of positive instances during training. As the number of epochs increases, the model is learned the pattern more efficiently and accurately and it is used to improve the recall. Recall (%), which is used to measures the overall correctly predictive positive instances. Table 7 presents Number of epoch Vs Recall. In Fig. 12 Number of epochs Vs Recall (%) is shown as a graph**.**

Table 7

Number of epochs Vs Recall (%)

X-axis (Number of epochs)	Y-axis (Recall (%))		
	Proposed	CADF	DL-GAN
20	90	82	70
40	92	84	72
60	94	86	74
80	96	88	76
100	98	89	78

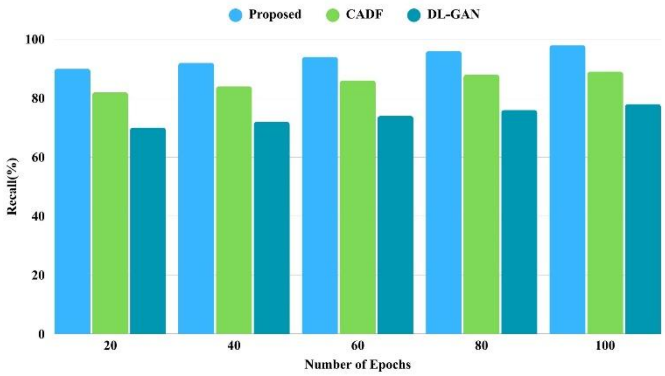

Fig. 12 Number of epochs Vs Recall (%)

The X-axis denotes the "Number of epochs" and Y-axis denotes the "recall", is used to identify the all-positive instances. Initially at 20 epochs, the proposed model at 90%, less than the CADF (82%) which shows that the smaller number of detecting the true positives. Finally at 100 epochs, CADF (89%) and DL-GAN (78%) decrease the recall whereas the proposed model achieves 98% of recall, which is demonstrating the detection of positive cases in superior.

C. Research Summary

In the proposed study, the experimental setup begins with the construction of the network environment that contains of fifty IoT devices, one macro base station, two micro base stations and central server for data management. Data is collected from the dataset and preprocessing using the Complex Event Processing (CEP) technique. CEP helps for aggregating, correlating raw data streams into the understandable information. After preprocessing the data, the feature selection is performed by using the "Binary Gravitational Search Algorithm (BGSA)" and "Binary Grey Wolf Optimization (BGWO)" with "Linear Discriminant Analysis (LDA)". The completion of feature selection process, the Long Short Term Memory model is used for analysing the network traffic patterns and identify the cyber-attack. The Grid Search Cross Validation are used to trained, then the "Synthetic Minority Oversampling Technique (SMOTE)" is used to balance the dataset used to generate the synthetic samples of minority attack. After, the classification is

performed using the a “Generative Adversarial Network (GAN)” integrated “Pigeon-Inspired Optimization (PIO) algorithm” with the Tanh activation function. It provides the effective detection of intruders both in internal and external by using optimizing decision. Finally, the performance is evaluated by using the metrics that has detection capabilities of the method. Graphs are plotted to shown the relationship between the “number of epochs” and evaluate the different measures, such as “Accuracy (%)”, “Precision (%)”, “Recall (%)”, “F1-score (%)”, “Loss (%)”, and “Detection Rate (%)”.

7. Conclusion

The proposed cyber threat detection framework that incorporates “pigeon- inspired Optimization (POI)” with “Generative Adversarial Networks (GAN)” and “SYN-GAN-PIA-TH” is used to enhance the security in the IoT environments. By combing grid-based network partitioning, Complex event processing (CEP), and the BGSA-BGWO-LDA hybrid feature selection, the proposed model effectively overcomes traditional limitation Boundary weakness, in accurate preprocessing, and inefficient feature extraction. The incorporation of Long Sort Term Memory (LSTM) networks enables accurate temporal analysis of network traffic, while GSCV-SMOTE ensures balanced learning for minority attack classes, reducing bias and false alarms. Overall, the proposed framework establishes a robustness, adaptive resource solution for next generation of IoT networks, capable of accurately detect the internal and external intrusions in real time. The “Experimental results” confirm that the “proposed system” achieves good performance in terms of “accuracy, detection rate, precision, recall and F1-score” compared with the existing model like CADF and DL-GAN. In future, the proposed model can be extended using the federated based learning.

Reference

1. Aminu, M., Akinsanya, A., Dako, D. A., & Oyedokun, O. (2024). Enhancing cyber threat detection through real-time threat intelligence and adaptive defense mechanisms. International Journal of Computer Applications Technology and Research, 13(8), 11-27.
2. Bakhsh, S. A., Khan, M. A., Ahmed, F., Alshehri, M. S., Ali, H., & Ahmad, J. (2023). Enhancing IoT network security through deep learning-powered Intrusion Detection System. Internet of Things, 24, 100936.
3. Park, C., Lee, J., Kim, Y., Park, J. G., Kim, H., & Hong, D. (2022). An enhanced AI-based network intrusion detection system using generative adversarial networks. IEEE Internet of Things Journal, 10(3), 2330-2345.
4. Sood, T., Prakash, S., Sharma, S., Singh, A., & Choubey, H. (2022). Intrusion detection system in wireless sensor network using conditional generative adversarial network. Wireless Personal Communications, 126(1), 911-931.
5. Xu, W., Jang-Jaccard, J., Liu, T., Sabrina, F., & Kwak, J. (2022). Improved bidirectional gan-based approach for network intrusion detection using one-class classifier. Computers, 11(6), 85.
6. Le, K. H., Nguyen, M. H., Tran, T. D., & Tran, N. D. (2022). IMIDS: An intelligent intrusion detection system against cyber threats in IoT. Electronics, 11(4), 524.
7. Aldhaheri, S., & Alhuzali, A. (2023). SGAN-IDS: Self-attention-based generative adversarial network against intrusion detection systems. Sensors, 23(18), 7796.
8. Liu, Z., Hu, J., Liu, Y., Roy, K., Yuan, X., & Xu, J. (2023). Anomaly-based intrusion on IoT networks using AIGAN-a generative adversarial network. IEEE Access, 11, 91116-91132.
9. Balaji, S., & Narayanan, S. S. (2023). Dynamic distributed generative adversarial network for intrusion detection system over internet of things. Wireless Networks, 29(5), 1949-1967.
10. Zhao, X., Fok, K. W., & Thing, V. L. (2024). Enhancing network intrusion detection performance using generative adversarial networks. Computers & Security, 145, 104005.
11. Lent, D. M. B., da Silva Ruffo, V. G., Carvalho, L. F., Lloret, J., Rodrigues, J. J., & Proença, M. L. (2024). An unsupervised generative adversarial network system to detect ddos attacks in sdn. IEEE Access, 12, 70690-70706.
12. Hossain, M. A., & Islam, M. S. (2023). Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. Array, 19, 100306.
13. Lavanya, P., Singh, R. P., Kumaran, U., & Kumar, P. (2024). Gradient boosting classifier performance evaluation using Generative Adversarial Networks. Procedia Computer Science, 235, 3016-3024.
14. Li, Z., Wang, P., & Wang, Z. (2024). Flowgananomaly: Flow-based anomaly network intrusion detection with adversarial learning. Chinese Journal of Electronics, 33(1), 58-71.
15. Saikam, J., & Ch, K. (2024). EESNN: Hybrid deep learning empowered spatial–temporal features for network intrusion detection system. IEEE Access, 12, 15930-15945.
16. Strickland, C., Saha, C., & Zakar, M. (2023). DRL-GAN: A Hybrid approach for binary and multiclass network intrusion detection. arXiv preprint arXiv: 230103368.
17. Maddu, M., & Rao, Y. N. (2024). Network intrusion detection and mitigation in SDN using deep learning models. International Journal of Information Security, 23(2), 849-862.
18. Javeed, D., Saeed, M. S., Adil, M., Kumar, P., & Jolfaei, A. (2024). A federated learning-based zero trust intrusion detection system for Internet of Things. Ad Hoc Networks, 162, 103540.

19. Mari, A. G., Zinca, D., & Dobrota, V. (2023). Development of a machine-learning intrusion detection system and testing of its performance using a generative adversarial network. Sensors, 23(3), 1315.
20. Kadry, H., Farouk, A., Zanaty, E. A., & Reyad, O. (2023). Intrusion detection model using optimized quantum neural network and elliptical curve cryptography for data security. Alexandria Engineering Journal, 71, 491-500.
21. Ilyas, M. U., & Alharbi, S. A. (2022). Machine learning approaches to network intrusion detection for contemporary internet traffic. Computing, 104(5), 1061-1076.
22. Liu, M., Yang, Q., Wang, W., & Liu, S. (2024). Semi-supervised encrypted malicious traffic detection based on multimodal traffic characteristics. Sensors, 24(20), 6507.
23. Chuang, H. M., & Ye, L. J. (2023). Applying transfer learning approaches for intrusion detection in software-defined networking. Sustainability, 15(12), 9395.
24. Ullah, F., Ullah, S., Srivastava, G., & Lin, J. C. W. (2024). IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic. Digital Communications and Networks, 10(1), 190-204.
25. Li, Z., Ge, Y., Guo, J., Chen, M., & Wang, J. (2022). Security threat model under internet of things using deep learning and edge analysis of cyberspace governance. International Journal of System Assurance Engineering and Management, 13(Suppl 3), 1164-1176.
26. Kumar, P., Gupta, G. P., & Tripathi, R. (2021). Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for iot networks. Arabian Journal for Science and Engineering, 46(4), 3749-3778.
27. Alomiri, A., Mishra, S., & AlShehri, M. (2024). Machine learning-based security mechanism to detect and prevent cyber-attack in IoT networks. International Journal of Computing and Digital Systems, 16(1), 645-659.
28. Thirimanne, S. P., Jayawardana, L., Yasakethu, L., Liyanaarachchi, P., & Hewage, C. (2022). Deep neural network based real-time intrusion detection system. SN Computer Science, 3(2), 145.
29. Imanbayev, A., Tynymbayev, S., Odarchenko, R., Gnatyuk, S., Berdibayev, R., Baikenov, A., & Kaniyeva, N. (2022). Research of machine learning algorithms for the development of intrusion detection systems in 5G mobile networks and beyond. Sensors, 22(24), 9957.
30. Kandhro, I. A., Alanazi, S. M., Ali, F., Kehar, A., Fatima, K., Uddin, M., & Karuppayah, S. (2023). Detection of real-time malicious intrusions and attacks in IoT empowered cybersecurity infrastructures. IEEE Access, 11, 9136-9148.