Telecommunication Systems (2024) 87:605–624
https://doi.org/10.1007/s11235-024-01200-y

A lightweight optimized intrusion detection system using machine learning for edge-based IIoT security

Ravi Shekhar Tiwari[1] · D. Lakshmi[2] · Tapan Kumar Das[3] · Asis Kumar Tripathy[3] · Kuan-Ching Li[4]

Accepted: 16 July 2024 / Published online: 26 July 2024

Abstract
The Industrial Internet of Things (IIoT) attributes to intelligent sensors and actuators for better manufacturing and industrial operations. At the same time, IIoT devices must be secured from the potentially catastrophic effects of eventual attacks, and this necessitates real-time prediction and preventive strategies for cyber-attack vectors. Due to this, the objective of this investigation is to obtain a high-accuracy intrusion detection technique with a minimum payload. As the experimental process, we have utilized the IIoT network security dataset, namely WUSTL-IIOT-2021. The feature selection technique Particle Swarm Optimization (PSO) and feature reduction techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed stochastic neighbor embedding (t-SNE) are applied. Additionally, the Generalized Additive Model (GAM) and Multivariate Adaptive Regression Splines (MARS) are used to detect payloads that can interfere with the normal operation of an application. Both PSO and PCA combined with MARS have produced predictive results with an exceptional accuracy of 100%. Yet, the trained Machine Learning (ML) model is quantized with 4-bit and 8-bit, and it is deployed on Azure IoT Edge to simulate edge devices. Experimental results show that the latency of the model was reduced by 25% on quantization.

Keywords Network intrusion detection · Industrial internet of things · Machine learning · PSO · PCA · MARS · Quantization

1 Introduction

The Industrial Internet of Things (IIoT) constitutes a vast range of industrial sensors, machinery, supporting applications, data centers, supporting operations, services, actuators, and people. IoT integration into industrial automation is known as IIoT, sometimes Industry 4.0 [1, 2]. Globalization has boosted internet usage in industries, businesses, and other disciplines worldwide. COVID-19 has caused a huge increase in the number of internet users. Therefore, designing reliable and secure apps is a key responsibility for the agile development cycle of any IT organization. Rapid application development presents various obstacles. Deploying a dependable network is a laborious but crucial process, which might lead to multiple security risks in the respective application [3].

Denial of Service (DOS) attacks are probably the most prevalent and straightforward attacks that overwhelm or flood a system with unwanted requests from bots. Yahoo was the initial target of DOS's first public attack in 2000 [4]. Currently, DOS targets web services, social media sites, and other web applications to saturate and overwhelm their resources with unsolicited requests [5] so that when a legitimate user arrives, the program cannot serve their demands. R2L is referred to as Remote-to-Local attacks. R2L is designed to have the local authorization for Remote-to-Local attacks against server resources that are only accessible to local users or are available exclusively on the intranet, such as file servers. These types of assaults, SPY and PHF, are under the umbrella of R2L attacks that try to gain unauthorized access to a network and its resources [6, 7]. U2R refers to Root

Ravi Shekhar Tiwari, D. Lakshmi, Tapan Kumar Das and Asis Kumar Tripathy have contributed equally to this work.

✉ Kuan-Ching Li
kuancli@pu.edu.tw

[1] Department of Computer Science Engineering, Mahindra University, Hyderabad, India

[2] School of Computing Science and Engineering, VIT Bhopal University, Bhopal, India

[3] School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore, India

[4] Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan

attacks that illegally modify the user privileges to Root-user privileges and have complete access to the computer and its network resources [8]. For attackers, it is a challenge to keep their tools up-to-date and add originality with authentication while exploiting any network or application vulnerability. It is exceedingly difficult for the IT department to detect assaults based on a single rule because attackers constantly modify their techniques to find application vulnerabilities.

Intrusion Detection System (IDS) plays a crucial role in network security. IDS's main function is to monitor real-time traffic and generate alerts when an attack is detected. IDS can be installed to monitor the entire network's traffic, referred to as N-IDS (Network-IDS), or to monitor the traffic of a single device, referred to as H-IDS (Host-IDS). IDS detects web applications for harmful activities. IDS is intended to detect known attacks and unusual actions, notify administrators, and prevent unauthorized access. IDSs are classified into Network-based IDSs (NIDS) and Host-based IDSs (HIDS). NIDSs analyze network traffic for malicious activity patterns, whereas HIDSs monitor the actions of a particular host or system. Signature-based detection is used by NIDSs, which compare incoming network traffic to a database of known attack patterns. This form of IDS detects known attacks well, but it is easily circumvented by attackers who utilize new or modified attack patterns. An alternate approach is Anomaly Intrusion Detection (A-IDS), which employs machine learning algorithms to identify anomalous behavior [9, 10]. This method is more effective in detecting unknown threats, although it has a large false-positive rate. HIDSs are normally deployed on individual servers or computers, and they can detect known and undiscovered assaults. These systems employ a mix of signature-based and anomaly-based detection methods and heuristics, which are rules and patterns used to detect suspicious behavior.

Based on the type of detection, IDS can be classified as either anomaly-based or misuse-based. Anomaly Intrusion Detection (A-IDS) is utilized to detect network traffic based on irregular payload traffic behavior. It compares the real-time traffic payload to the sample payload recorded in its database [11, 12]. A-IDS is widely used and provides the highest level of security. However, it generates many false positive alerts since unseen packages are interpreted as network attack payloads.

Misuse Intrusion Detection (M-IDS) is implemented to identify threats based on the signatures of similar payloads in its repository. Because the load is not recorded in the warehouse, the new attack signature can pass through a pass-through detection system without triggering an alarm. Currently, M-IDS is unreliable because the traffic payload signature might vary substantially based on the user or host network [13].

1.1 Motivation for the study

In the contemporary digital age, network traffic analysis in the IIoT (Industrial Internet of Things) is an increasingly essential topic for companies and businesses. As the number of linked devices grows, so does the potential for cyber criminals to disrupt operations; organizations must use effective intrusion detection techniques to identify cyberattacks. Detection of network attacks is considered a classification challenge [14] because payload signatures can be classified as either normal or attack payloads. Consequently, the Machine Learning (ML) model may be utilized to classify the payload signature throughout the network. Since network payload might contain n-numbers of data, such as Source IP, Destination IP, and other information linked to the network and requests being delivered to the serving application, these are referred to as features. In this paper, we propose a novel method for feature selection called PSO, which selects elements based on the impact on the dependent variable. Subsequently, the MARS and GAM classifier models are employed to choose features based on the threshold of the feature contribution specified by the end-user when training the model.

1.2 Objectives

The aims of this research work are listed as:

- How to detect the payloads originating from the hacker's machine to the application server with a minimum latency
- How to reduce the computational cost of the machine learning algorithm for the detection of the cyberattack
- How to block a specific IP address or user request following the Standard Operating Procedure (SoP) of WWW
- How do you provide the best predictive accuracy for the imbalanced dataset? In general, the cyberattack data will be imbalanced.

The current study of developing an intrusion detection system has few significant contributions. They are:

- We proposed an IDS for identifying attacks.
- We investigated PSO, PCA, LDA, and tSNE feature selection/reduction techniques connecting with MARS and GAM classifiers to recognize the best combination.
- The proposed system achieved an accuracy of 100% over IIoT network traffic data
- The trained ML model is simulated in the Azure IOT platform with reduced response time.

The remainder of the paper is structured as follows: Sect. 2 covers the review of literature pertinent to network intrusion detection and classification. The detailed methodology

is described in Sect. 3. In Sect. 4, we discussed the experiment's results and analyzed the model evaluation metrics. The paper concludes in Sect. 5.

2 Related studies

The size, complexity, and data of networks have increased due to recent internet and network communication developments. Numerous components and a sophisticated communication network and architectural system are vulnerable to various cyberattacks and attack vectors [15, 16]. Agarwal *et al.* [17] discussed a few factors that are tough to monitor for an IDS to identify web attacks. Additionally, they highlighted available technologies to monitor web traffic based on their design and functionality.

The major reasons for the network threats are:

- The complex system increases the possibility of misconfigurations and flaws that lead to unintended access
- Attackers become familiar with the common code and existing system loopholes
- Interconnected devices via ports
- Poor and unregulated password management

The reasons above necessitate a dynamic and efficient Intrusion detection system (IDS). Studies show that ML and Deep Learning (DL) techniques also provide promising solutions for detection and prevention. NIDS is one of these, and it is the attack detection mechanism that, through continuously scanning network traffic for hostile and suspicious activity, provides greater protection [18, 19].

Kim and Pak [20] proposed a classification method based on LSTM-DNN and a GAN-based validation model to detect network intrusions in real-time. Rani *et al.* [21] proposed detecting DDoS attacks in device-to-device communication utilizing the CICDDoS2019 standard dataset in their experimentation. Four different types of ensemble models are being used in this experiment. The highest accuracy is achieved with random forest ensemble classifiers. Rustam *et al.* [22] proposed a method of classification to predict normal DoS attacks and DDoS attacks using the Kaggle dataset. They used four different models inclusive of linear and nonlinear, an ensemble of homogeneous base learners, and an ensemble of heterogeneous base learners. They achieved 100% accuracy with random forest ensemble classifiers.

Zolanvari *et al.* [23] proposed SCADA-IDS testbed data to reduce the number of false negatives. There are seven different machine learning algorithms used. Almost all seven algorithms have resulted in 90%-99% accuracy. Othman *et al.* [24] proposed a comparative analysis using a Chi-SVM classifier and a Chi-Logistic Regression classifier using the KDD99 big data IDS data set. The Spark-Chi-SVM model resulted in high accuracy. Ganapathy *et al.* [25] proposed an intelligent feature selection technique and a rule-based enhanced multiclass support vector machine using KDD'99 Cup data. The accuracy of the multi-label classification has increased with the proposed method compared to other traditional methods. Wang [26], in his study, compares the performance of cutting-edge attack techniques versus deep learning-based intrusion detection on the NSL-KDD data set using the Multilayer Perceptron (MLP) classifier. The vulnerability of the intrusion detection system is tested experimentally using a neural network. Individual features and their significance in generating adversarial examples are investigated. The viability and usefulness of the attack tactics are examined based on their findings. Akhtar and Feng [27] used the NSLKDD dataset and developed a novel binary and multiclass classification model to discover network system anomalies using Convolutional Neural Networks (CNNs).

Lansky *et al.* [28] surveyed several host- and network-based IDSs developed and tested at various institutions. The host-based IDSs rely primarily on the audit trails of the host operating system to detect intrusion, while most network-based IDSs monitor network traffic. Some network-based IDSs also use host audit trails as a source of input. The paper overviews a statistical anomaly detection algorithm commonly used in IDSs. Divyasree and Sherly [29] present an efficient intrusion detection system that detects U2R, R2L, Probe, and DoS attacks based on the Ensemble Core Vector Machine (ECVM) approach.

Singh *et al.* [30] generated synthetic data using Monte Carlo simulation to generate the wireless sensor network for essential synthetic predictors. The entire machine-learning prediction is automated using AutoML.

Liu *et al.* [31] present an intrusion detection technique using a PSO-based Gradient Descent (PSO-LightGBM). Furthermore, it has been shown that PSO, when used as feature selection, has a reduced false positive rate compared to the random forest algorithm used in feature selection, depicted by Kunhare *et al.* [32].

It is important to make cutting-edge Network Intrusion Detection Systems (NIDS) powered by artificial intelligence (AI) to protect IIoT networks from increasingly complex and varied attacks. Many of the algorithms reported in the literature have achieved 98% to 99% accuracy. Some algorithms used deep learning approaches, whereas a few utilized machine learning approaches. There is a lack of effective feature selection and extraction techniques, so the IIoT network may provide 100% accuracy in detecting intrusion and a reduced computational cost.

3 Methodology

3.1 Dataset

In this investigation, we utilized the WUSTL-IIoT 2021 dataset,[1] consisting of IIoT traffic network data created by IIoT testbed sensors [13, 23]. Its main goal is to replicate the real-world situation of cyberattacks. The dataset is intentionally designed to resemble a real-world scenario and is imbalanced. The network was subjected to command injection attacks, surveillance, and denial of service against the IIoT testbed [33] with an average data rate of 419 kb/sec and an average payload size of 76.76 bytes. Figure 1 depicts the proportion of various types of attacks present in the dataset.

The dataset contains 1,194,464 samples and 49 feature sets, of which 87,016 samples are labeled as 1, i.e., attacked traffic, and 1,107,448 samples are labeled as 0, i.e., normal traffic.

3.2 Dataset preprocessing

The WUSTL-IIOT 2021 dataset has 49 features, of which 9 are categorical characteristics and 40 are numerical features. The researchers noted that some feature sets, including "StartTime," "LastTime," "SrcAddr," "DstAddr," "sIpId," and "dIpId" present in the dataset, need to be excluded because they are specific to attacks and could result in a data leakage scenario while training the model. Therefore, we must consider the remaining feature set from the dataset, i.e., 43 columns (42 features and one target variable). The data was preprocessed by eliminating missing values, invalid entries, and outliers before making it public.

There are 1,194,464 samples in the dataset, of which 95,557 samples, or 8% of the total dataset, are attack/intrusion traffic samples, and 10,98,907 samples, or 92% of the total dataset, are normal traffic samples. The researchers intentionally created the imbalance dataset to simulate a real-world scenario. There were 42 feature sets in the dataset after the six feature sets, "StartTime," "LastTime," "SrcAddr," "DstAddr," "sIpId," and "dIpId," which were present in the dataset, needed to be eliminated [33]. The numerical columns were subjected to the Min-Max scaling algorithm [30] to normalize the characteristics. The data's min-max scaling makes it simple for a machine learning model to learn and predict. The Min-Max scaler is a method of displaying numerical numbers between 0 and 1. The Min-Max scale is defined mathematically in the eq. 1

$$x_i = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

[1] https://www.cse.wustl.edu/jain/iiot2/index.html.

Table 1 Pre-processed dataset information

Total samples	1,194,464
Attack traffic	87,016
Normal traffic	1,107,448
Total number of features	49
Number of features after preprocessing	42

where x_i is the scaled feature value of the i_{th} row, x_{min} is the minimum value of the respective column, and x_{max} is the maximum value from the respective columns. Min-max scaler is an alternative to Z-score normalization; it gives us the values that are less scattered, i.e., smaller standard deviation, thus removing the concept of the outlier from the dataset [34–36]. Table 1 shows the dataset information after preprocessing.

3.3 Proposed methodology

The dataset was perfectly suited to demonstrate the effectiveness of PSO for picking the features iteratively based on the influence on the dependent variables, as it included 42 features and 1,194,464 total samples. The block diagram of the proposed methodology is exhibited in Fig. 2.

The methodology consists of the following major steps:

- The preprocessing of the dataset involves scaling the numerical features, one-hot encoding of the categorical variables, outlier detection, and missing value detection
- PSO technique is used as a feature selection method; PCA- an unsupervised and linear model, is used for dimensionality reduction; LDA- a supervised and linear model, is employed for dimensionality reduction; and lastly, tSNE- an unsupervised and nonlinear model, is used for the dimensionality reduction
- Subsequently, GAM and MARS classifiers are trained on the feature set based on the threshold value for PSO, and the best components or lower dimensions are used for the dimensionality reduction techniques
- Lastly, the classification model is validated and analyzed with accuracy, F1- score, and ROC metrics

3.4 Feature selection/feature reduction

In machine learning and data analysis, a subset of pertinent features is selected to incorporate into a model or algorithm. By minimizing complexity and noise in the data, feature selection aims to isolate the most crucial aspects that influence prediction or decision-making. This can enhance the models' functionality, speed, and interpretability, reduce overfitting, and enhance generalization to new data. There are numerous ways to choose features, including wrapper, filter,

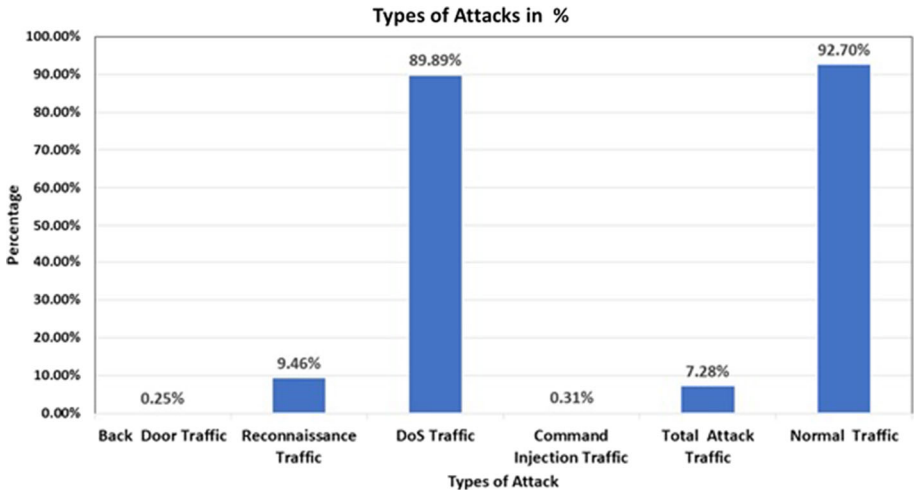

Fig. 1 Types of attack

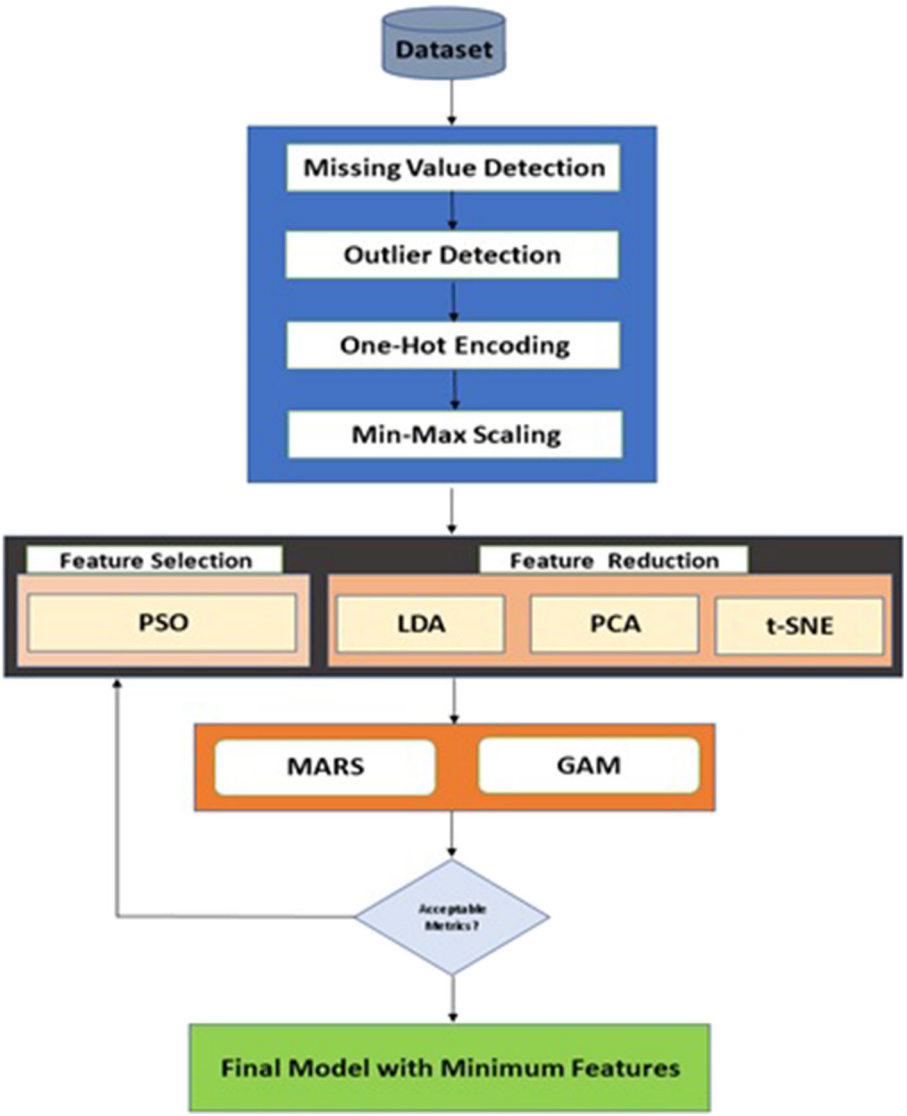

Fig. 2 Proposed Methodology exhibiting the classifiers and feature extractors

and embedded techniques, each of which has its advantages and disadvantages. Building powerful and effective machine learning models requires understanding and using feature selection.

Feature reduction is a data analysis technique that includes eliminating or merging features to decrease the dimensionality of a data set. The objective of feature reduction is to preserve critical information while reducing redundant, unnecessary, or noisy characteristics from the data. This can enhance the effectiveness and speed of ML models while, in turn, making them more understandable and appealing. There are several techniques for feature reduction, including principal component analysis, singular value decomposition, and linear discriminant analysis. It is crucial to comprehend and use feature reduction when preprocessing data to create ML models that are both effective and efficient.

Data dimensionality turns into a computational curse. The technique for choosing the prominent characteristics or removing the undesirable features is known as feature selection. However, dimensionality reduction reduces the data dimensions without losing any significant information. There are a few advantages of dimensionality reduction: 1) the efficiency and accuracy can be improved to a larger extent by feature selection or feature reduction; 2) the overfitting can be avoided; 3) the data leakage can be avoided; 4) the computation time can be reduced. We have utilized dimensionality reduction algorithms such as Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), and Linear Discriminant Analysis (LDA) to extract the most vital features from the dataset that affect the dependent variable. In our study, we employed the MARS classification algorithm in combination with PSO, PCA, LDA, and t-SNE. Furthermore, the GAM model is also used with the same feature selection/reduction techniques.

3.4.1 Particle swarm optimization (PSO)

PSO is used to solve optimization issues in science and engineering. It is an optimization technique that mimics how a bunch of particles would behave in a search space where each particle would represent a potential solution. To locate the overall optimal solution, the particles move around the search space and modify their placements based on their own and their neighbors' experiences. PSO is a versatile and effective optimization method that can solve various optimization problems. Each bird in PSO is referred to as a particle, and when we think about them all at once, we refer to them as swarms. The region in which they are searching is referred to as hyperspace. According to the location of the food, the individual bird closest to it, known as p_best, provides input that is used to modify the direction and position of the group of birds known as g_best. PSO is an optimization technique that has been around for around 70 years and is utilized in various areas of computing [37].

For selecting the features that affect the dependent variable in our proposed study, the main goal is to choose the best feature set from the existing features. The PSO algorithm began by selecting features at random from the available features. Based on feedback from the classifier, it has updated the individual and global best feature sets and added them to a list of the most sensitive features. Finally, it has updated the velocity of the PSO based on test set model feedback, where is the parameter. We have explicitly chosen the value of to be 0.5, though its actual range is between 0 and 1. If we want to give weightage to the test set model feedback, $if\ (a = 0)$, then the velocity value does not depend on the test set feedback of the model. To reach the features threshold fitness, the entire process is performed a predetermined number of times. The suggested approach can be used with any classifier.

The PSO algorithm and its involving operations are explained in Algorithm 1.

Algorithm 1 Algorithm for feature selection

For each epoch:
Select random features from the available feature set
Store the target variable and store it in dataset_y
Split-up the dataset into Train and Test Set in ratio of 80:20
Fit the Model with the selected label and target variable
if fit > fitness[$i-1$] **then**
  if length$_{p_{\text{best}}} \leq$ LEN (selected$_{\text{features}}$) **then**
    $p_{\text{best}} = \text{selected}_{\text{features}}$
    length$_{p_{\text{best}}}$ = LEN (selected$_{\text{features}}$)
  end if
end if
if $g_{\text{best}} \leq$ fit **then**
  $g_{\text{best}} = \text{selected}_{\text{features}}$
end if
if $length_{p_{best}} <= LENGTH\left(selected_{features}\right)$ **then**
  Test the model with the test data
  $\text{test}_{\text{Factor}} = \alpha \times \text{testfit}$
  $\text{group}_{\text{factor}} = c_2 * \text{rand}() * (g_{\text{best}} - \text{fit})$
  $\text{individual}_{\text{factor}} = c_i * \text{rand}() * (p_{\text{best}} - \text{fit})$
  $\text{Velocity} = \text{Velocity} + \text{test}_{\text{Factor}} + \text{group}_{\text{factor}} + \text{individual}_{\text{factor}}$
  distance = distance + Velocity
  fitness.append(fit)
end if

By employing the above algorithm on the experimental dataset, we have drastically reduced the number of features from 42 to 10. Figure 3 shows the feature contribution for the dependent variable. Feature names are represented along the X-axis, and the Y-axis denotes the feature importance on a scale of 0-1, where 1 represents the highest contributor and 0 is the lowest contributor.

Our dataset has 42 features. However, when we statistically analyze the feature set from the graph in Fig. 3, PSO has statistically demonstrated that most of the features have contributions equal to zero and that 10 out of the 42 features

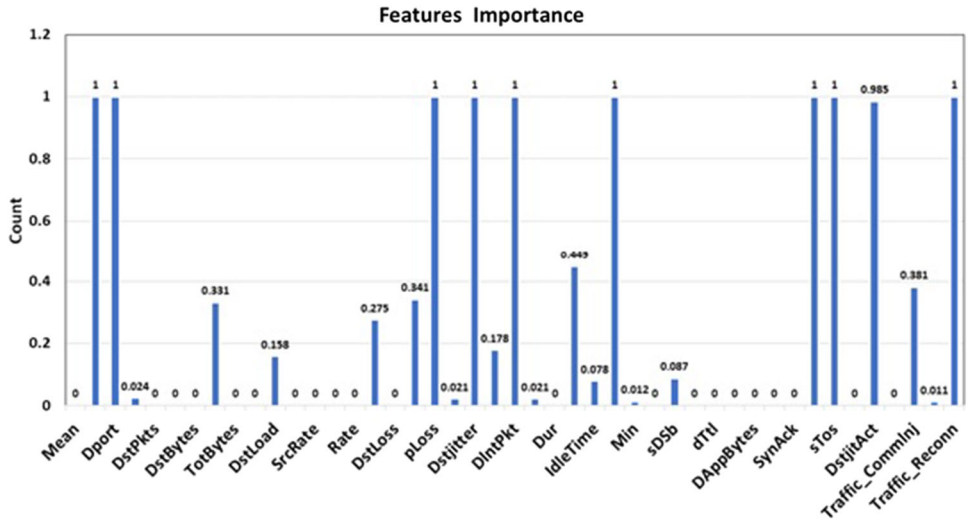

Fig. 3 Feature contribution of all features

are primarily contributing to the prediction. In contrast, the rest have contributions of less than 50% or 0.5. Because PSO will have less latency and concurrently execute, it will significantly impact when we deploy the model in real-time for intrusion detection. We will illustrate how we trained and tested our model using the extracted features based on the threshold in the following section. MARS and GAM models were iteratively trained on the feature set by adjusting the feature threshold starting from an upper bound of 1 and lower bound of 0.5 and lowering the threshold by 0.1 in each iteration so that the number of features varies based on its contribution to the model prediction.

The deployment of the ML model with ten extracted features is represented in Fig. 4. The significant features determined by PSO are Dport, Srcpkts, Srcjitter, Slntpkts, Proto, Min, Runtime, Stos, Idletime, and Dstjitact. The ML model is trained only by these selected parameters. Once the model is trained, it can be deployed for classifying attacked and normal traffic.

3.4.2 Principal component analysis (PCA)

PCA is one of the multivariate techniques and works as a linearly separable agenda. This method is pursued as a matrix solvation of eigenvalue and eigenvector problems. When many variables are used in a dataset, PCA retains the variations as much as possible. The dimensionality reduction is achieved by a new set of features called uncorrelated principal components. These components preserve most of the variation of the given dataset. PCA is an unsupervised learning model aimed to find the direction that maximizes the variance in the dataset. The PCA algorithm [38] is presented in Algorithm 2.

The principal components chosen are 10, 9, 8, 7, 6, 5, 4 and 3. All the principal components have resulted in 100% accuracy. The binary classification with the five principal components (PC1, PC2, PC3, PC4, PC5) is shown visually in Fig. 5.

Algorithm 2 Algorithm for feature selection

1: Calculating the mean value for each feature:

$$F_{\text{mean}_i} = \frac{d_1 + d_2 + d_3 + \ldots + d_n}{n}$$

where F_{mean_i} is the feature mean for $i \in \{1, m\}$, and m is the number of features. $d_1 + d_2 + d_3 + \ldots + d_n$ are data points, and n is the size of the vector.

2: Standardizing each feature of the dataset using the feature mean. Center the data at the origin:

3: Calculate Covariance matrix:

$$\text{Variance:} \quad \text{var}(x) = \frac{\sum_i^n (x_i - \mu)}{n}$$

$$\text{Covariance:} \quad \text{Cov}(x_i, x_j) = \frac{\sum_i^n (x_i - \mu_i)(x_j - \mu_j)}{n}$$

4: Eigenvalues and Eigenvectors:

$$(M - \lambda I_n)\mathbf{v} = 0$$

where M is an $n \times n$ matrix, λ is a scalar, and $\mathbf{v}$ is a vector.

5: Sorting the eigenvalues in descending order: The highest represents the principal component.

6: Form a feature vector and transform the original dataset:

3.4.3 Linear discriminant analysis (LDA)

An approach for supervised learning called LDA looks for the linear combination of features that maximizes class separability. LDA aims to decrease the feature space's dimensionality while maintaining class discriminability. Finding the linear discriminant functions that maximize class separability is how LDA is carried out. First, the mean vectors of the features for each class are computed, followed by the

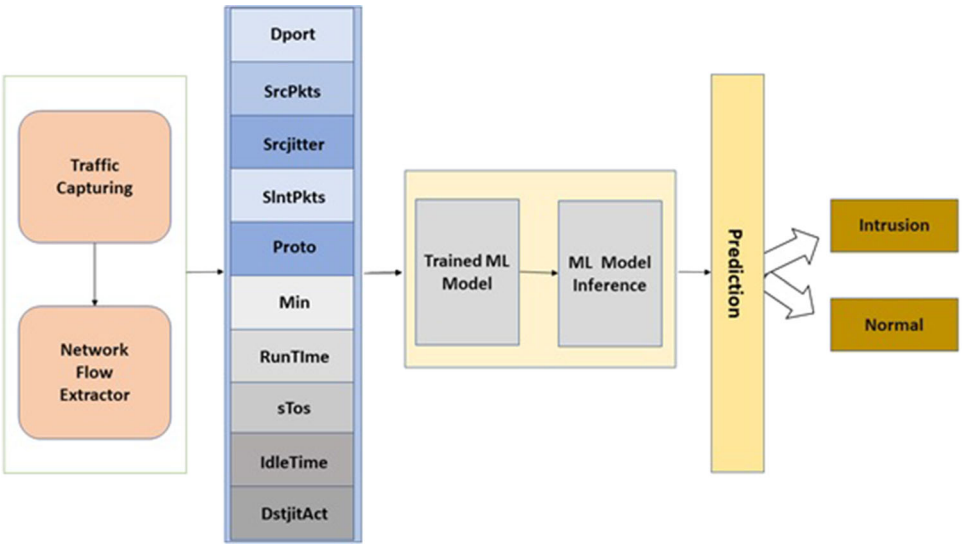

Fig. 4 Machine learning model with optimized parameters

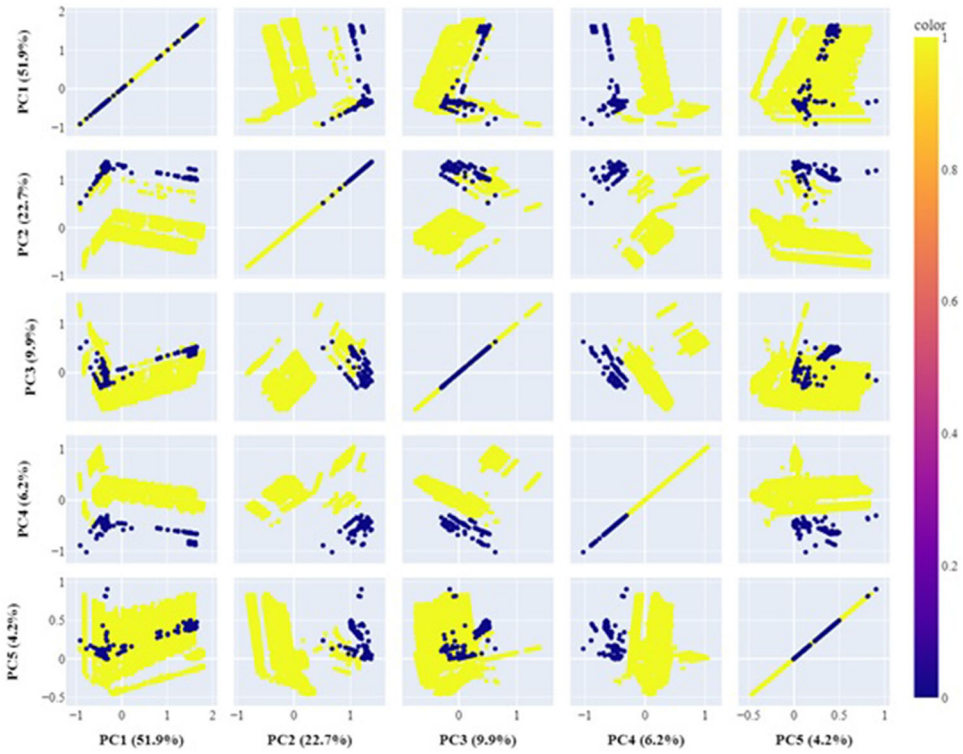

Fig. 5 Binary classification with five principal components

covariance matrix of the features. LDA is useful for both the classification of multi-labels and for dimensionality reduction as well. It reduces the number of independent variables before fitting to the predictive models. LDA is a supervised learning model that finds the directions to represent the axes that maximize the class separation. In general, the dimensionality reduction helps reduce the computation cost and helps to avoid overfitting. The data shape used in the MARS classification after the dimensionality reduction using LDA is (176438, 1). The LDA algorithm [39–41] is presented in Algorithm 3.

Algorithm 3 Linear Discriminant Analysis technique

1: Computation of mean vector for each feature:
2: Computing inter-class scatter matrix
3: Computing intra-class scatter matrix
4: Finding the eigenvalues and eigenvectors:
5: $(M - \lambda I_n)\mathbf{v} = 0$ where, M is an $n \times n$ matrix, λ is a scalar, and $\mathbf{v}$ is a vector.
6: Eigen vector matrix is used to transform the samples into new subspaces

3.4.4 t-Distributed stochastic neighbor embedding (t-SNE)

In the area of data visualization, the t-SNE model has gained popularity. The t-SNE can handle nonlinear manifold structures. It also helps with dimensionality reduction and feature extraction. This model consists of three key steps: computing similarity in higher and lower dimensions and reducing the difference between them. The objective of t-SNE is to preserve the structure and relationships between data points while mapping high-dimensional data into a lower-dimensional space. Building a probability distribution across pairs of data points is the operative mechanism of t-SNE.

The k-best approach, which is based on the Chi-square method, is used for dimensionality reduction employing t-SNE. Out of all the features, $(n = 2)$ performs better. The feature engineering process employs the Select Best K approach. It uses a single variable to choose features. It operates by choosing the top attributes using univariate statistical analyses. This procedure can be thought of as an estimator's preprocessing stage. The t-SNE algorithm calculates the probability score for the similarity of original data points in high-dimensional space and also computes the probability of similarity of data points in the corresponding low-dimensional space. In Fig. 5, the low dimensional space of the WUSTL-IIOT-2021 dataset is represented. The difference between conditional probabilities will be minimized by summing the Kullback-Leibler divergences of all data points. The t-SNE algorithm [42, 43] is presented in Algorithm 4.

Algorithm 4 Visualizing high dimensional data in 2D maps

1: Finding pairwise similarity among the original points in a high-dimensional space:
2: Calculate $P_{j|i}$ as a higher dimensional probability.
3: Similarity points are calculated as a conditional probability based on Euclidean distance
4: where $i \neq j$.

$$P_{j|i} = \frac{e^{-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}}}$$

5: Similarly, calculate q_{ij} as the lower dimensional probability

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|\right)^{-1}}{\sum_k \sum_{l \neq k} k \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

6: Compute the cost function using Kullback-Leibler divergence

$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

The data shape after the feature extraction (176438, 1, 2, 3, 4, 5, 6) is shown visually in Fig. 6. The figure is a two-dimensional plot that represents the high-dimensional data in a compressed format. Each point on the plot represents a data point, and the position of the point reflects the similarity between the data points. In a t-SNE graph, similar data points are grouped in clusters, classified into normal traffic and attack traffic, while dissimilar points are separated. The distance between the clusters of normal traffic and attack traffic reflects their dissimilarity. The size of the clusters reflects the density of data points in that area.

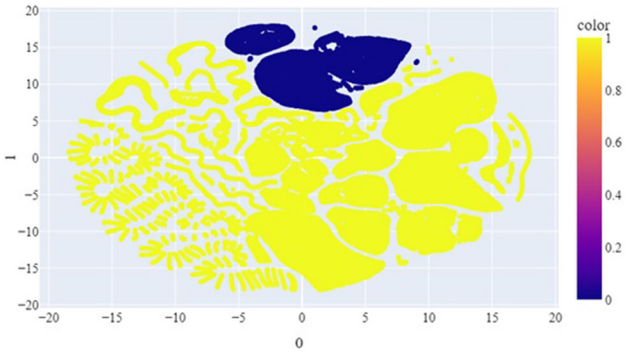

Fig. 6 t-SNE dimensionality reduction visualization

3.5 Classification algorithms

3.5.1 Multivariate adaptive regression splines (MARS)

This method includes the built-in capacity to discover any nonlinearity existing in the data and identify interactions between the variables. To get the best answer, it searches for all feasible models. This algorithm works well for regression, classification, and feature selection [44]. It performs well both as a stand-alone approach and also combined with other approaches as well. The MARS model involves two major steps: the growing phase, or forward step, where terms are added to the model, and the pruning phase, or backward step, where terms are deleted from the model. The MARS algorithm [45] is explained in Algorithm 5:

3.5.2 Generalized additive model (GAM)

GAMs are a class of statistical models that extend the generalized linear model (GLM) by incorporating non-linear relationships between the response and predictor variables, using smooth functions instead of linear functions as used by linear models. The following is pseudocode for fitting a GAM:

1. For a set of predictor variables, select a response variable Y
2. Choose a link function α and a probability distribution f for Y

Algorithm 5 MARS Algorithm

1: Define a linear function $f(x) = \sum k_{i=1} c_i B_i(x)$ where $c_i = Linear constant\ and\ B_i = Basis function$
2: Each $B_i(x)$ can take any of the below forms:
   (i) An intercept
   (ii) A hinge function, i.e., max (0, x-constant) or max (0, x+constant)
   (iii) A product of two or more hinge functions
3: The process involves two steps:
   (i) Forward Pass
      - Start with intercept, i.e., null model
      - Add $B_i(x)$ in pairs to the model; pairs are mirrors of one another
      - Explore each existing term, variable, and value of variables
   (ii) Backward Pass
   (iii) Terms are removed one by one based on Generalized Cross Validation (GCV). where $GCV = \frac{1}{N}\sum_{i=1}^{N} \frac{(y_i - f)^2}{\left(1 - \frac{ENP}{N}\right)^2}$ ENP is an effective number of parameters and $ENP = k + d(k-l)/2$ where k = number of MARS term and d = penalty term

3. Define the linear predictor $\eta = \beta_0 + f_1(\beta_1) + f_2(\beta_2) + f_3(\beta_3) + \ldots + f_n(\beta_n)$
4. Transform η to the scale of Y using the link function $\alpha : \mu = \alpha(\eta)$
5. Choose a suitable loss function L to estimate the parameters: $\beta_0, f_1(\beta_1), f_2(\beta_3), \ldots f_n(\beta_n)$
6. Estimate the smooth functions $f_1(\beta_1) + f_2(\beta_2) + f_2(\beta_3) + \ldots + f_n(\beta_n)$ that minimizes the loss function L
7. Maximize the likelihood function $L(\beta_0, f_1(\beta_1), f_2(\beta_2), f_2(\beta_3), \ldots f_n(\beta_n))$ to estimate the parameters $\beta_0, f_1(\beta_1), f_2(\beta_2), f_2(\beta_3), \ldots f_n(\beta_n)$
8. Compute Y using $\mu = \alpha(\eta)$

3.6 Model quantization

Quantization is a technique used in ML and deep learning(DL) to reduce the memory and computational requirements of neural network models while maintaining their performance. It involves converting the model's weights and sometimes activations from high-precision floating-point numbers (e.g., 32-bit) to lower-precision fixed-point numbers (e.g., 8-bit integers). This reduction in precision can significantly decrease the memory footprint and computation time, making it easier to deploy ML/DL models on resource-constrained devices such as smartphones, embedded systems, and IoT devices. Post-training quantization (PTQ) is a technique used to reduce the memory and computational requirements of ML/DL models after they have been trained with standard high-precision floating-point numbers. In 8-bit quantization, model weights and activations are converted into 8-bit integers, significantly reducing memory usage and computational overhead while minimizing the loss of accuracy. However, 8-bit quantization requires careful selection of quantization ranges to avoid saturation issues or excessive loss of precision. In the case of 4-bit quantization, the precision is further reduced to just 4 bits, resulting in a more substantial compression but a more considerable drop in accuracy. The challenge with 4-bit quantization is to strike the right balance between compression and model performance. Both 8-bit and 4-bit PTQ can be particularly valuable when deploying ML/DL models on resource-constrained devices while maintaining acceptable levels of model accuracy, making them essential techniques for edge computing and embedded systems. The trade-off is that lower-precision models may not perform as well as their high-precision counterparts, but optimizing the quantization process can help minimize this performance loss.

4 Results and discussions

The validation metrics used are the confusion matrix, Precision (P), recall (R), accuracy(A), Receiver Operating Curve (ROC), and the F1- score (F1) to compare the performance of all four different approaches. After a certain number of features or principal components, the classification accuracy is stagnant or insignificant. The summary of classification model validation metrics such as accuracy and F1 score are tabulated in Table 2.

4.1 PSO-MARS versus PSO-GAM

The key findings and overall performance of the PSO with the MARS and GAM classification model are represented in Fig. 7. The performance evaluation metrics used to evaluate this algorithm include AUC, ROC, F1, P, R, and A. The MARS classification model with PSO employed five unique threshold values (0.9, 0.8, 0.7, 0.6, and 0.5) based on the features of importance. All five models yielded outstanding results with a perfect accuracy rate of 100%, which is the best outcome that has been achieved.

GAM with PSO also achieved analogous results of 100% for F1, P, R, and A for various thresholds like its counterpart MARS technique. Hence, both the models, when operated with PSO, achieved the maximum performance.

Top- 10 features from 42 features were chosen using PSO. The MARS classification model was then iteratively trained using the best top features from 10, 9, 8, 7, 6, and 5, with the number of features varied according to the importance threshold for each feature. The maximum accuracy with the fewest features was what we were targeting.

Table 2 Summary of model validation measures

Feature Selection / Feature Reduction	Selected Features	Accuracy	F1 Score
PSO	Features of importance are assigned having threshold values of 0.9, 0.8, 0.7, 0.6, and 0.5	100	100
PCA	Selected number of principal components are 10, 9, 8, 7, 6, 5, 4, 3	100	100
LDA	Two components are chosen for the prominent features	98	98
t-SNE	Single component is chosen for the prominent features	88	88

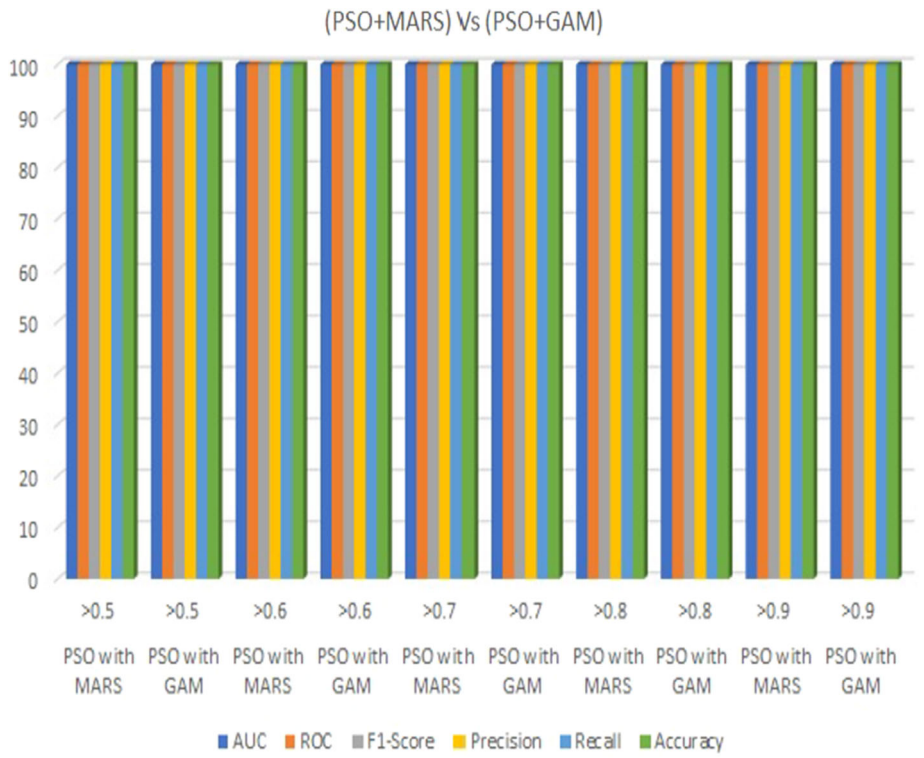

Fig. 7 PSO with MARS and GAM: validation metrics

4.2 PCA-MARS versus PCA-GAM

The key findings and overall performance of the PCA-MARS classification and PCA-GAM model are shown in Fig. 8. In the MARS model from PCA, the selected number of principal components are 10, 9, 8, 7, 6, 4, and 3. All seven models yielded outstanding results with a perfect accuracy rate of 100%. The best outcome has been achieved using a minimum of 3 principal components. That reduces the entire computational cost on the larger dataset.

From Fig. 8, the maximum outcomes are attained by considering a minimum of 3 components. Hence, it is concluded that PCA delivers identical performance with either of the classification techniques.

We used the PCA dimensionality reduction technique on a dataset to find the principal components ranging from 0 to 10 exclusively, and we iteratively trained the MARS algorithm by changing the number of feature components from PCA.

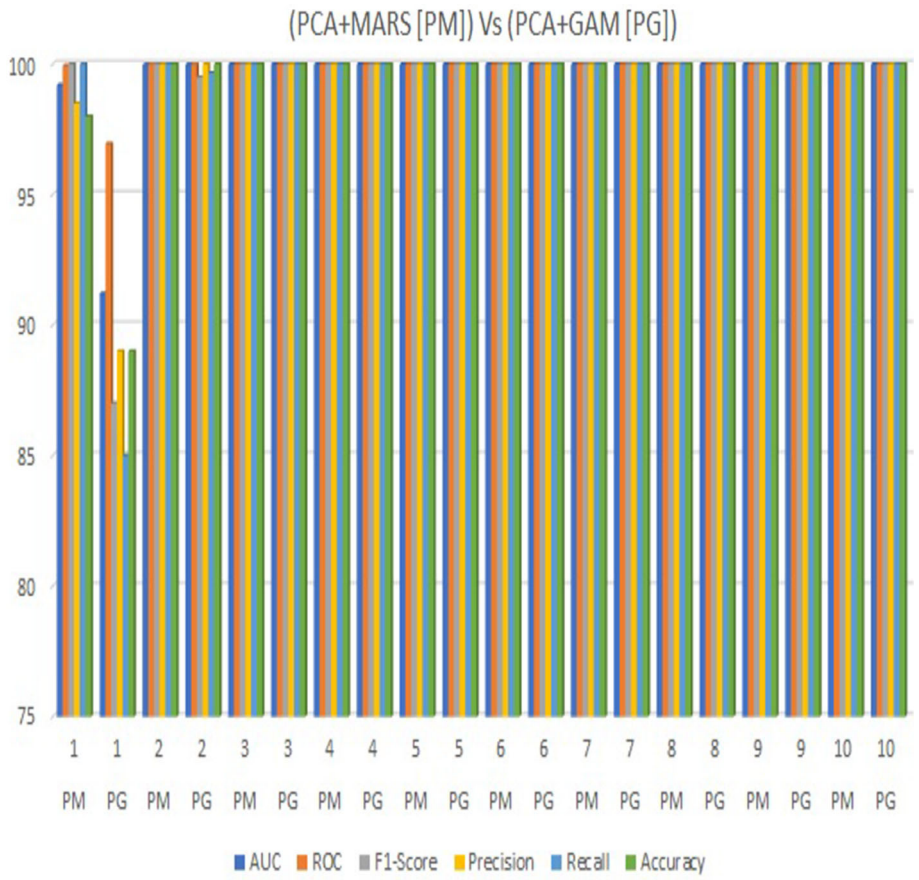

Fig. 8 PCA with MARS and GAM algorithm: validation metrics

4.3 LDA-MARS versus LDA-GAM

The key findings and overall performance of the LDA-MARS classification model are exhibited in Fig. 9. The MARS classification model from LDA has two components based on the features of importance. The model has yielded the results with an accuracy rate of 98%.

As compared to MARS, GAM has a slight edge as it produces an accuracy of 100% and attains ROC of 100% also. It is represented in Fig. 10. We used the LDA dimensionality reduction technique on a dataset to find the prominent features ranging from 0 to 10 exclusively, and we iteratively trained the MARS algorithm by changing the number of features from LDA.

4.4 tSNE-MARS versus tSNE-GAM

The key findings and overall performance of the t-SNE with the MARS classification model are shown in Fig. 11. The MARS model from t-SNE has one component based on the features of importance. The model yielded the results with an accuracy rate of 88%.

From Fig. 12, it is obvious that the GAM model optimized with t-SNE generated an accuracy of 95%, which is much higher as compared to MARS. Other parameters such as AUC, ROC, and precision are also better values than MARS. Hence, it is revealed that the GAM model is undoubtedly the best combination with t-SNE. We employed the tSNE method to reduce the dimensions of our dataset and identify the most notable features within a range of 0 to 10. The MARS algorithm was trained iteratively by adjusting the number of features from tSNE.

By using both the PSO feature selection and PCA dimensionality reduction as a preprocessing technique, the MARS classification algorithm and GAM model produced promising prediction accuracy and an F1 score of 100

4.5 MARS classifier with quantization

The trained MARS classifier is quantized with 4-bit and 8-bit quantization. The accuracy of the quantized model is represented in Fig. 13. From the Figure, it is revealed that the accuracy of MARS+PSO before quantization was 100%, however. On 8-bit quantization, it is slightly reduced to 99.16%, whereas on 4-bit quantization it is 98.09%. The same trend is noticed with PCA, LDA, and t-SNE.

The Response time vs Quantization graph of MARS is exhibited in Fig. 14. It is pertinent from the figure that the

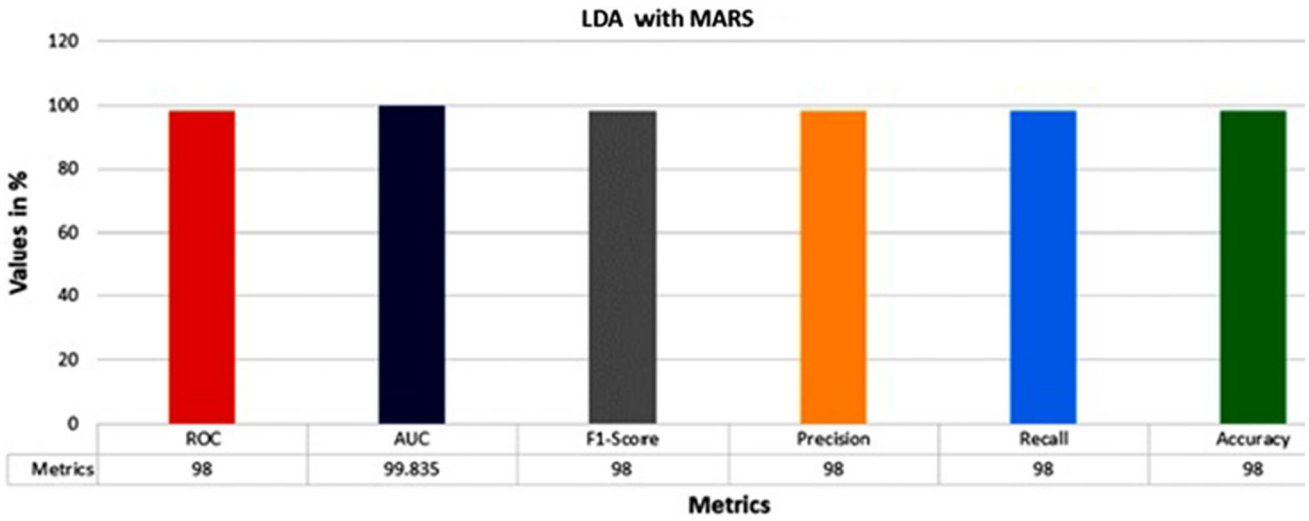

Fig. 9 LDA with MARS algorithm: validation metrics

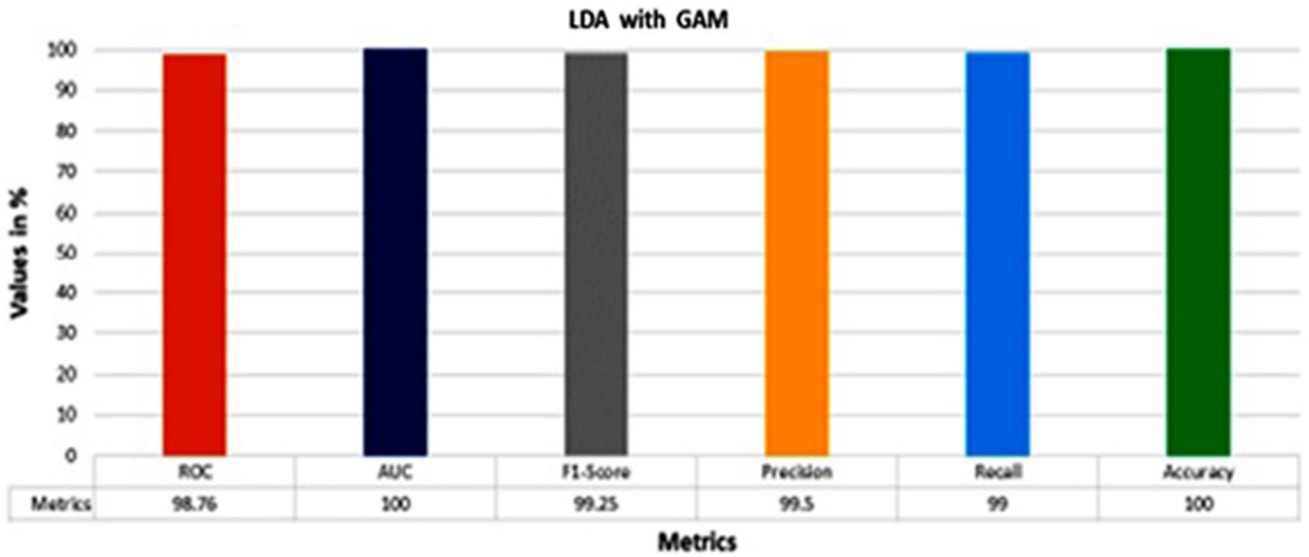

Fig. 10 LDA with GAM model: validation metrics

response time of the MARS+PSO reduces by 25% on 8-bit quantization while reducing by 22% on 4-bit quantization.

4.6 GAM classifier with quantization

The trained GAM classifier is also quantized for 8-bit and subsequently for 4-bit. The testing results of the GAM classifier for accuracy and response time are represented in Fig. 15 and Fig. 16, respectively. Clearly, from the figure, the model's accuracy is nominally reduced by 1 to 3% due to quantization. However, the response time of GAM combined with PSO was reduced by 25%, and GAM with PCA was reduced by 44%.

The detailed result of both the classifiers on quantization is represented in Table 3. The Table reveals that latency on quantization decreased for both the classifiers combined with either of the four feature selection/reduction techniques.

4.7 Simulation on IoT azure

Simulating edge devices on Azure and deploying ML models onto them is a crucial step in developing and deploying edge computing solutions. Azure provides a comprehensive suite of services and tools to facilitate this process. To begin, we create virtual IoT devices on Azure IoT Edge, which mimic real edge devices. These virtual devices allow testing and fine-tuning of ML models in a controlled environment before deployment. The simulation framework of our ML model is represented in Fig. 17.

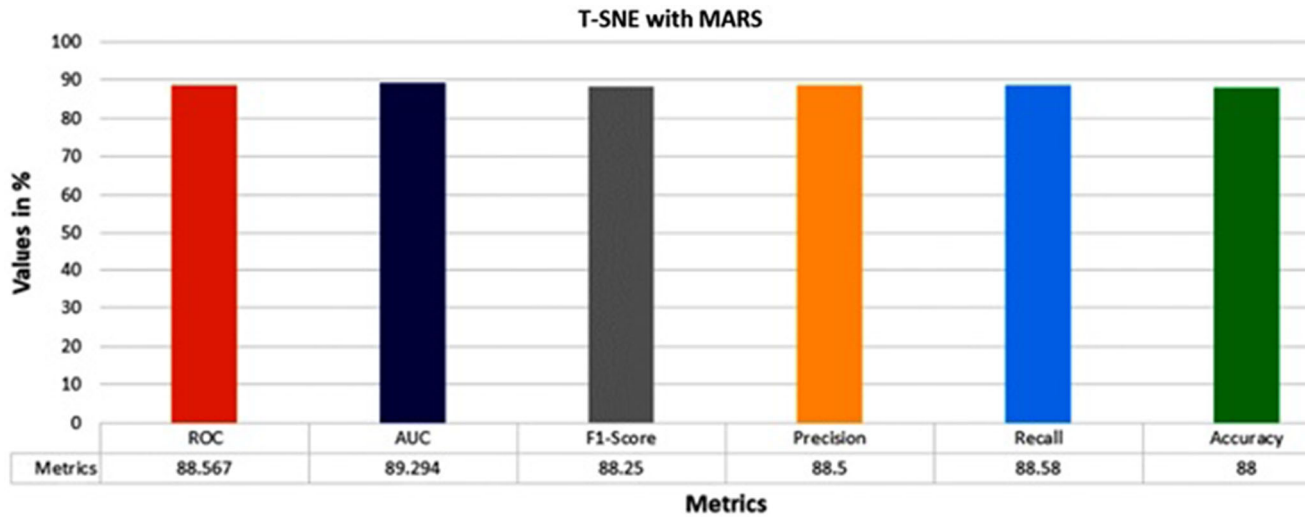

Fig. 11 t-SNE with MARS algorithm: validation metrics

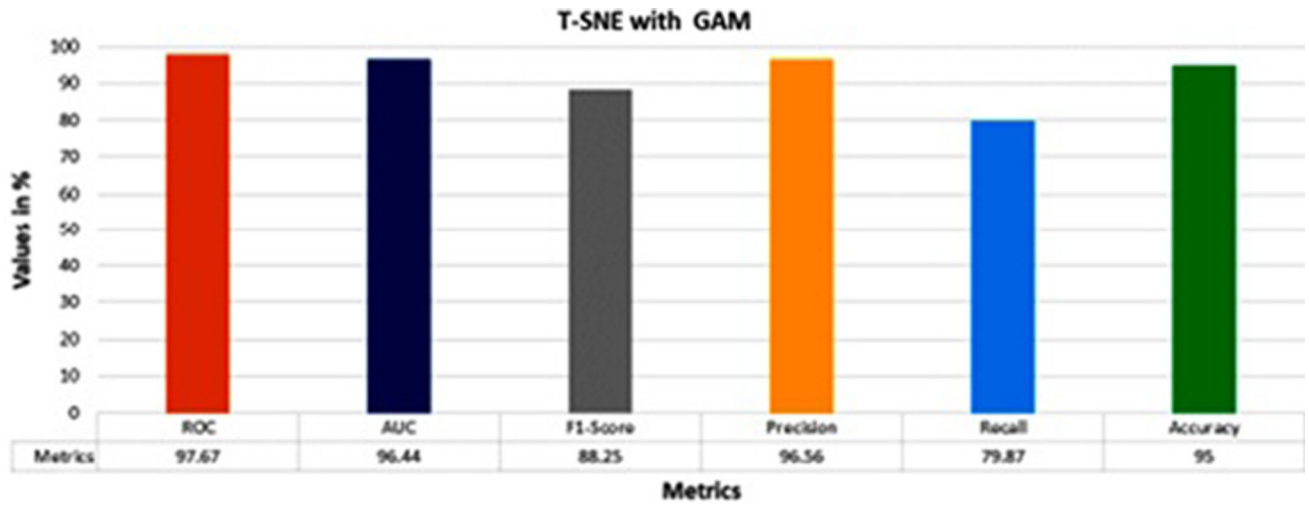

Fig. 12 t-SNE with GAM model: validation metrics

Table 3 Performance comparison of the model

Model Name	Feature Selection/ Reduction	Latency with Quantization Techniques (sec)			Accuracy with Quantization Techniques (%)		
		8-Bit	4-Bit	Without Quantization	8-Bit	4-Bit	Without Quantization
GAM	PSO	8.94	6.8	9.04	99.04	98.76	100
GAM	PCA	12.0	8.16	14.55	86.27	85.02	89
GAM	LDA	9.35	8.67	11.79	98.55	97.25	100
GAM	T-SNE	14.8	15.4	16.56	86.76	84.76	86
MARS	PSO	7.04	6.8	8.98	99.16	98.06	100
MARS	PCA	12.48	11.12	13.05	98.03	96.01	100
MARS	LDA	8.24	7.26	9.37	96.25	94.95	78
MARS	T-SNE	12.55	10.58	15.25	87.06	85.86	88

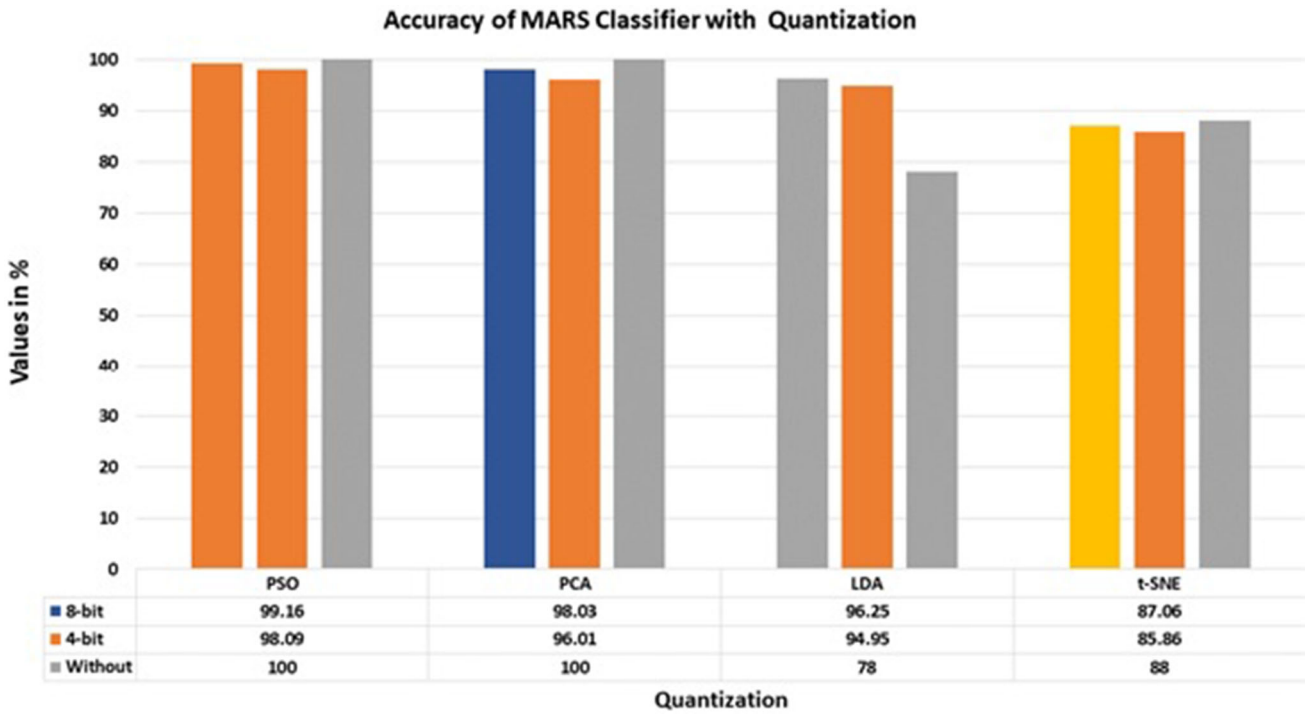

Fig. 13 Accuracy of MARS classifier with quantization vs without quantization

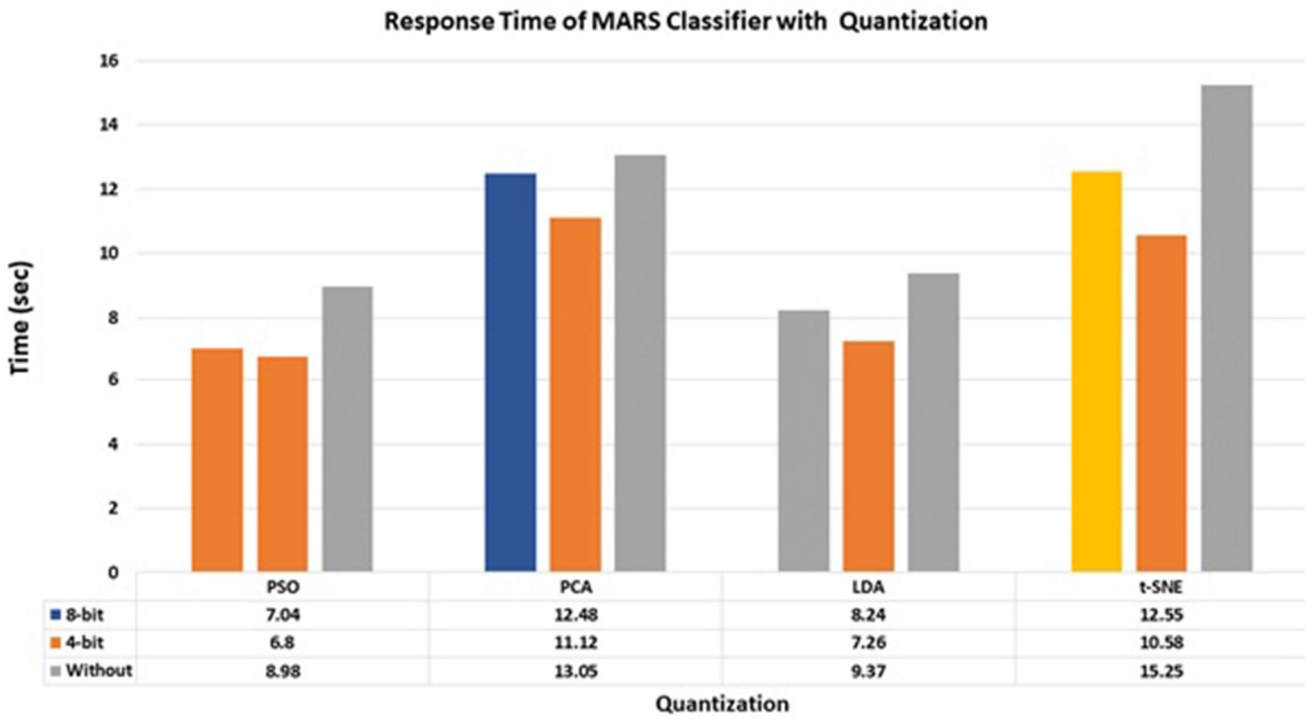

Fig. 14 Response time of MARS classifier with quantization vs without quantization

After quantization, we deployed the proposed model on Azure IoT Edge to simulate edge devices; we configured the deployment settings of the IoT edge device to Raspberry Pi 3 Model B+, which features a quad-core ARM Cortex-A53 CPU, which provides enhanced performance compared to its predecessor. It has 1GB of LPDDR2 RAM, four USB 2.0 ports, a microSD card slot for storage, and a single HDMI output supporting resolutions up to 1080p. The Model B+

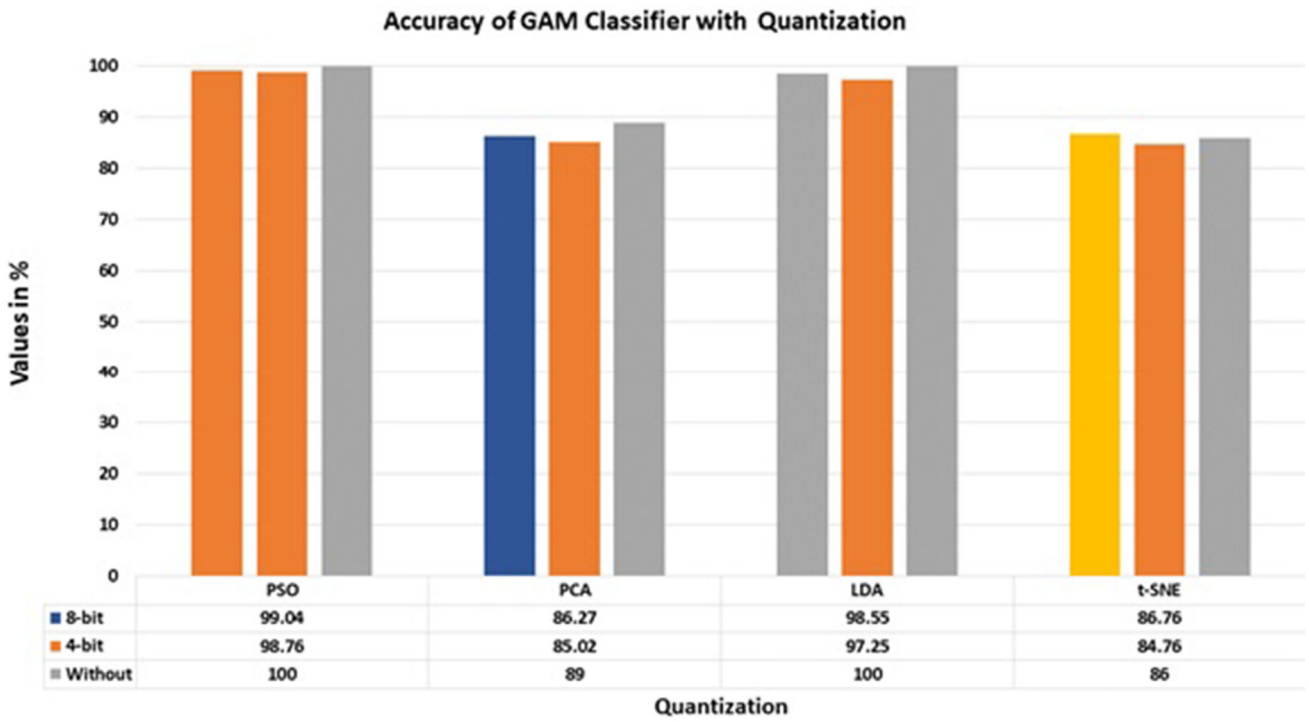

Fig. 15 Accuracy of GAM classifier with quantization vs without quantization

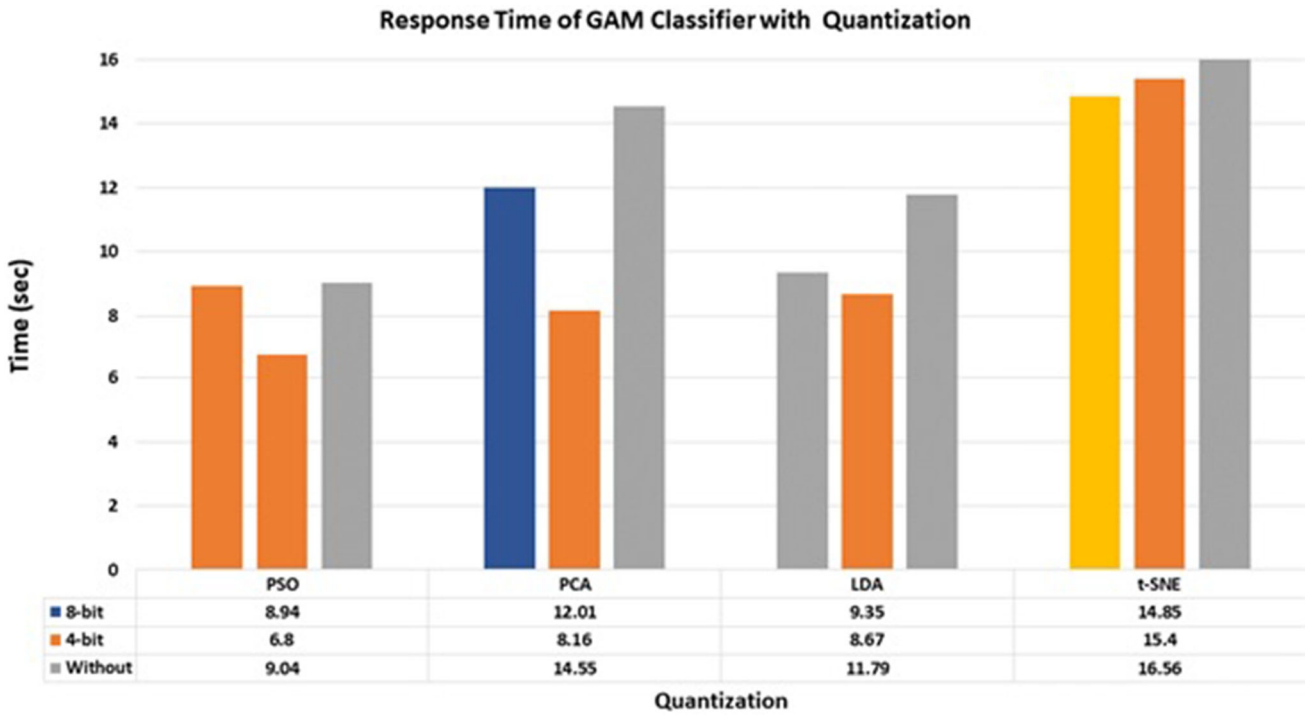

Fig. 16 Response time of GAM classifier with quantization vs without quantization

Table 4 Comparison of our results with existing studies'

Studies	Methodology	Result		
		Accuracy	Sensitivity	False Positive Rate
Zolanvari et al. [[23]	Naïve Bayes	97.48	97.44	2.52
	Decision Trees	99.98	94.87	0.01
	KNN	99.98	91.03	0.01
	Random Forest	99.99	97.44	0.01
	Logistic Regression	99.9	47	0.01
	SVM	99.64	65.38	0
Alani et al. [[46]]	MLP	99.94	99.92	0.01
Zolanvari et al. [[33]]	TRUST XAI	99.98	–	–
Proposed Model	PSO + MARS (>0.5)	100	100	0
	PSO + MARS (>0.6)	100	100	0
	PSO + MARS (>0.7)	100	100	0
	PSO + MARS (>0.8)	100	100	0
	PSO + MARS (>0.9)	100	100	0
	LDA + MARS (component=1)	98.76	98.88	0.407
	t-SNE + MARS (components=2)	88	84	3.62
	PCA + MARS (component=1)	98.97	98	100
	PCA + MARS (components=2)	99.93	99	100
	PCA + MARS (components=3)	100	100	100
	PCA + MARS (components=4)	100	100	100
	PCA + MARS (components=5)	100	100	100
	PCA + MARS (components=6)	100	100	100
	PCA + MARS (components=7)	100	100	100
	PCA + MARS (components=8)	100	100	100
	PCA + MARS (components=9)	100	100	100
	PCA + MARS (components=10)	100	100	100

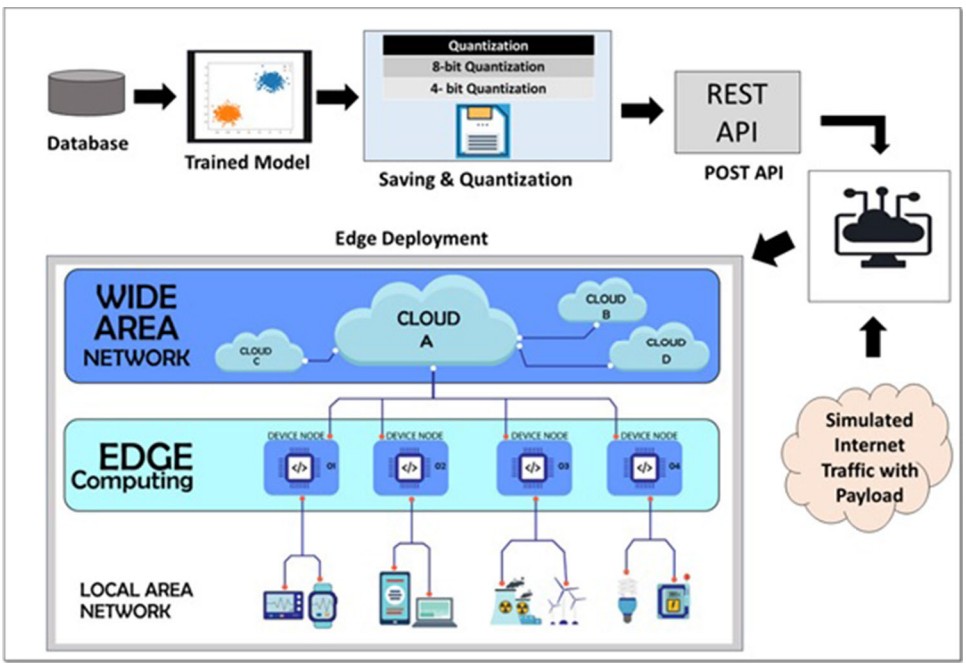

Fig. 17 Simulation framework of the proposed model

offers 10/100 Ethernet and 802.11n Wi-Fi, along with Bluetooth 4.2 support for networking. We are ensuring that our model runs efficiently in an edge-computing context. Azure IoT Hub complements this by providing device management and telemetry monitoring, allowing us to track the simulated edge devices and their performances.

4.8 Comparison with existing studies

In this section, we presented a comparison between the results of our proposed study and the existing literature that utilized the WUSTL-IIOT 2021 dataset for their experimentation. The comparison is briefed in Table 4.

From the above analysis, out of all the models in the past, the RF model generated an accuracy of 99.99%, which was assumed to be the best one for the given data set. However, our results reveal 100% accuracy in most of the setups, such as PSO +MARS and PCA+MARS. Furthermore, GAM optimized with PSO also achieved 100% accuracy.

5 Conclusions and future work

The utility of feature reduction, feature selection, and machine learning models has been investigated in this study for detecting attacked traffic in the industrial IoT network. The findings demonstrated that the combination of these methodologies can efficiently pinpoint and foresee possible network weaknesses, offering a complete remedy for enhancing the security of IIoT systems. The feature space size can be decreased by using feature reduction and feature selection techniques, leading to a lightweight intrusion detection system. From the experiment, both PSO and PCA, in combination with MARS or GAM machine learning models, PSO and PCA achieved 100% accuracy in classifying attacked traffic from normal ones. The combination of PSO and MARS is superior: a) Faster real-time predictions resulting in lower latency, and b) consider only highly relevant features in network traffic, as the number of payloads in a request varies. The trained model is also quantized and subsequently deployed in the IoT Azure environment for simulation. The advantage of the proposed system lies in outperforming other intrusion detection systems in terms of model accuracy and computation time, which is a critical factor in IIoT. The system also provides a good trade-off between effectiveness and efficiency. Furthermore, the proposed approach is apposite for attack detection and classification, particularly in a real-time IIoT environment. Hence, it can be used in real-time to mitigate attacks.

In future directions, resource consumption can be treated as additional evaluation parameters to enhance our studies of the proposed IIoT-IDS framework. This can include the analysis of energy consumption, memory utilization, and processing complexity using resource-aware IoT nodes with associated system elements. Additionally, the resiliency of IDS against adversarial attacks can also be tested.

Author Contributions Ravi Shekhar Tiwari: Methodology, Writing-Original draft preparation, Software; Lakshmi D: Conceptualization, Methodology, Writing- Original draft preparation, Resources, Supervision; Asis Kumar Tripathy: Methodology, Data curation, Resources, Software, Validation; Tapan Kumar Das: Methodology, Data curation, Validation, Visualization, Supervision, Writing- Reviewing and Editing; Kuan-Ching Li: Writing- Reviewing and Editing.

Funding The authors have not disclosed any funding.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Latif, S., Zou, Z., Idrees, Z., & Ahmad, J. (2020). A novel attack detection scheme for the industrial internet of things using a lightweight random neural network. *IEEE Access, 8*, 89337–89350.
2. Jhanjhi, N., Humayun, M., & Almuayqil, S. N. (2021). Cyber security and privacy issues in industrial internet of things. *Computer Systems Science & Engineering, 37*(3), 40.
3. Ye, Q., Bie, H., Li, K.-C., Fan, X., Gong, L., He, X., & Fang, G. (2021). Edgeloc: A robust and real-time localization system toward heterogeneous iot devices. *IEEE Internet of Things Journal, 9*(5), 3865–3876.
4. Kessler, G. C. (2012). *Denial-of-service attacks. Computer security handbook* (p. 18). New Jersey: Wiley.
5. Nguyen, H.A., & Choi, D. (2008). Application of data mining to network intrusion detection: classifier selection model. In: Challenges for Next Generation Network Operations and Service Management: 11th Asia-Pacific Network Operations and Management Symposium, APNOMS 2008, Beijing, China, October 22-24, 2008. Proceedings 11, pp. 399–408. Springer
6. Paliwal, S., & Gupta, R. (2012). Denial-of-service, probing & remote to user (r2l) attack detection using genetic algorithm. *International Journal of Computer Applications, 60*(19), 57–62.
7. Tripathy, P. I., Kumar, Asis, & Jena, D. (2010). Proxy blind signature based on ecdlp. *International Journal of Computer and Network Security, 2*, 1–7.
8. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A.A. (2009). A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications, pp. 1–6. Ieee
9. Zhang, S., Hu, B., Liang, W., Li, K.-C., & Pathan, A.-S.K. (2023). A trajectory privacy-preserving scheme based on transition matrix and caching for iiot. IEEE Internet of Things Journal
10. Tang, B., Guo, F., Cao, B., Tang, M., & Li, K. (2022). Cost-aware deployment of microservices for iot applications in mobile edge computing environment. IEEE Transactions on Network and Service Management
11. Han, D., Zhou, H., Weng, T.-H., Wu, Z., Han, B., Li, K.-C., & Pathan, A.-S.K. (2023). Lmca: a lightweight anomaly network traffic detection model integrating adjusted mobilenet and coordinate attention mechanism for iot. Telecommunication Systems, 1–16
12. Sethuraman, S. C., Mitra, A., Li, K.-C., Ghosh, A., Gopinath, M., & Sukhija, N. (2022). Loki: A physical security key compatible iot

based lock for protecting physical assets. *IEEE Access, 10*, 112721–112730.
13. Zolanvari, M., Teixeira, M.A., & Jain, R. (2018). Effect of imbalanced datasets on security of industrial iot using machine learning, in: 2018 IEEE international conference on intelligence and security informatics (ISI), pp. 112–117. IEEE
14. Amudha, P., Karthik, S., & Sivakumari, S. (2013). Classification techniques for intrusion detection-an overview. *International Journal of Computer Applications, 76*(16), 1042.
15. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies, 32*(1), 4150.
16. Gu, J., & Lu, S. (2021). An effective intrusion detection approach using svm with naïve bayes feature embedding. *Computers & Security, 103*, 102158.
17. Agarwal, N., & Hussain, S. Z. (2018). A closer look at intrusion detection system for web applications. *Security and Communication Networks, 2018*, 960135.
18. Li, J., Qu, Y., Chao, F., Shum, H. P., Ho, E. S., & Yang, L. (2019). Machine learning algorithms for network intrusion detection. *AI in Cybersecurity, 2019*, 151–179.
19. Prasad, R., Rohokale, V., Prasad, R., & Rohokale, V. (2020). Artificial intelligence and machine learning in cyber security. *Cyber Security: The Lifeline of Information and Communication Technology, 20*, 231–247.
20. Kim, T., & Pak, W. (2022). Early detection of network intrusions using a gan-based one-class classifier. *IEEE Access, 10*, 119357–119367.
21. Rani, S. J., Ioannou, I., Nagaradjane, P., Christophorou, C., Vassiliou, V., Charan, S., Prakash, S., Parekh, N., & Pitsillides, A. (2023). Detection of ddos attacks in d2d communications using machine learning approach. *Computer Communications, 198*, 32–51.
22. Rustam, F., Mushtaq, M. F., Hamza, A., Farooq, M. S., Jurcut, A. D., & Ashraf, I. (2022). Denial of service attack classification using machine learning with multi-features. *Electronics, 11*(22), 3817.
23. Zolanvari, M., Teixeira, M. A., Gupta, L., Khan, K. M., & Jain, R. (2019). Machine learning-based network vulnerability analysis of industrial internet of things. *IEEE Internet of Things Journal, 6*(4), 6822–6834.
24. Othman, S. M., Ba-Alwi, F. M., Alsohybe, N. T., & Al-Hashida, A. Y. (2018). Intrusion detection model using machine learning algorithm on big data environment. *Journal of Big Data, 5*(1), 1–12.
25. Ganapathy, S., Kulothungan, K., Muthurajkumar, S., Vijayalakshmi, M., Yogesh, P., & Kannan, A. (2013). Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. *EURASIP Journal on Wireless Communications and Networking, 2013*, 1–16.
26. Wang, Z. (2018). Deep learning-based intrusion detection with adversaries. *IEEE Access, 6*, 38367–38384.
27. Akhtar, M. S., & Feng, T. (2021). Deep learning-based framework for the detection of cyberattack using feature engineering. *Security and Communication Networks, 2021*, 1–12.
28. Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S. H. T., Rashidi, S., Hosseinzadeh, M., & Rahmani, A. M. (2021). Deep learning-based intrusion detection systems: a systematic review. *IEEE Access, 9*, 101574–101599.
29. Divyasree, T., & Sherly, K. (2018). A network intrusion detection system based on ensemble cvm using efficient feature selection approach. *Procedia Computer Science, 143*, 442–449.
30. Singh, A., Amutha, J., Nagar, J., Sharma, S., & Lee, C.-C. (2022). Automl-id: Automated machine learning model for intrusion detection using wireless sensor network. *Scientific Reports, 12*(1), 9074.
31. Liu, J., Yang, D., Lian, M., & Li, M. (2021). Research on intrusion detection based on particle swarm optimization in iot. *IEEE Access, 9*, 38254–38268.
32. Kunhare, N., Tiwari, R., & Dhar, J. (2020). Particle swarm optimization and feature selection for intrusion detection system. *Sādhanā, 45*, 1–14.
33. Zolanvari, M., Yang, Z., Khan, K., Jain, R., & Meskin, N. (2021). Trust xai: Model-agnostic explanations for ai with a case study on iiot security. IEEE internet of things journal
34. Jain, Y. K., & Bhandare, S. K. (2011). Min max normalization based data perturbation method for privacy protection. *International Journal of Computer & Communication Technology, 2*(8), 45–50.
35. Jain, Y., & Bhandare, S. (2013). Min max normalization based data perturbation method for privacy. *International Journal of Computer & Communication, 4*, 233–238.
36. Singh, B. K., Verma, K., & Thoke, A. (2015). Investigations on impact of feature normalization techniques on classifier's performance in breast tumor classification. *International Journal of Computer Applications, 116*(19), 44578.
37. Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence, 1*, 33–57.
38. Vasan, K. K., & Surendiran, B. (2016). Dimensionality reduction using principal component analysis for network intrusion detection. *Perspectives in Science, 8*, 510–512.
39. Balakrishnama, S., & Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. *Institute for Signal and Information Processing, 18*(1998), 1–8.
40. Xanthopoulos, P., Pardalos, P.M., Trafalis, T.B., Xanthopoulos, P., Pardalos, P.M., & Trafalis, T.B. (2013). Linear discriminant analysis. Robust data mining, 27–33
41. Tharwat, A., Gaber, T., Ibrahim, A., & Hassanien, A. E. (2017). Linear discriminant analysis: A detailed tutorial. *AI Communications, 30*(2), 169–190.
42. Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research, 9*(11), 14455.
43. Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to use t-sne effectively. *Distill, 1*(10), 2.
44. Quirós, E., Felicísimo, Á. M., & Cuartero, A. (2009). Testing multivariate adaptive regression splines (mars) as a method of land cover classification of terra-aster satellite images. *Sensors, 9*(11), 9011–9028.
45. Milborrow, S. (2019). Derived from mda: mars by trevor hastie and rob tibshirani. Uses Alan Miller's Fortran
46. Alani, M.M., Damiani, E., & Ghosh, U. (2022). Deepiiot: An explainable deep learning based intrusion detection system for industrial iot, in: 2022 IEEE 42nd international conference on distributed computing systems workshops (ICDCSW), pp. 169–174. IEEE

Ravi Shekhar Tiwari is a researcher, innovator, and an engineer. He has 4+ years of industry experience working as an Artificial Intelligence Engineer, Penetration Tester, and MFDI Engineer in Multinational IT companies as well as start-ups. He also holds a position as a reviewer and editor in reputed journals and as an author in technical magazines. His research domain includes Time Series Analysis, Protein Structure Prediction and Generation, Federated Learning, the Internet of Things, Microcontrollers, Gait Analysis, AI and Healthcare, XAI, Cloud Computing, Computer Vision, Parallel and Distributed Computing in the cloud. Currently, he is pursuing his Master in Technology at Mahindra University with Specialization Artificial Intelligence and Data Science.

D Lakshmi has 26 years of teaching experience. She is currently working as Senior Associate Professor in VIT Bhopal university. She has conducted FDPs that cover approximately 1,00,000 members. She has 20 international conference presentations and 2 papers that got best paper awards, 35 international journal papers inclusive of SCOPUS & SCI. A total of 20 SCOPUS-indexed book chapters. A total of 24 patents are in various states at national and international levels. She was appointed as an external examiner for several Ph.D candidates. Under her supervision, one scholar has completed a Ph.D. She received two educational awards in the year 2022. She has 3 books under self-publication. She acted as a SPOC to SIH-2022, SIH-2023, and KAVACH-2023. She has created 15 hours of the high-quality "Python Essentials" course at VITyarthi Platform. She has 7 SCOPUS Indexed book edits.

Tapan Kumar Das currently working as Professor in School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore. He has graduated from Berhampur University, received M. Tech. from Utkal University in 2003 and Ph.D. degree in Computer Science and Information Technology from VIT University, India in 2015. He has about 20 years of experience in various capacity in IT industry and academics. He has authored more than 50 journals & conference research articles, book chapters with various reputed publishers. His research interests include Artificial Intelligence, Machine Learning and Data Analytics, Medical informatics and cognitive computing. He is a senior member of IEEE, life Member of Computer Society of India and Indian Science Congress Association.

Asis Kumar Tripathy is a professor in the School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India. He has more than ten years of teaching experience. He completed his Ph.D. from the National Institute of Technology, Rourkela, India. His areas of research interests include wireless sensor networks, cloud computing, Internet of things and advanced network technologies. He has several publications in refereed journals, reputed conferences and book chapters to his credit. He has served as a program committee member in several conferences of repute. He has also been involved in many professional and editorial activities. He is a senior member of IEEE and a member of ACM.

Kuan-Ching Li is currently a Life Distinguished Professor at Providence University, where he is also the Director of the High- Performance Computing and Networking Center, established by collaborations with industry. Besides publishing articles in renowned journals and conferences, he is a co-author/co-editor of more than 50 books published by well-known publishers. His research interests include parallel and distributed computing, Big Data, and emerging technologies. Dr. Li is a Senior Member of the IEEE, a Fellow of the IET and a member of the AAAS.