Adaptive Intrusion Detection in Edge Computing Using Cerebellar Model Articulation Controller and Spline Fit

Kumar, Gulshan; Saha, Rahul; Conti, Mauro; Thomas, Reji; Devgun, Tannishtha; Rodrigues, Joel J.P.C.

DOI
10.1109/TSC.2022.3178471

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Services Computing

Citation (APA)
Kumar, G., Saha, R., Conti, M., Thomas, R., Devgun, T., & Rodrigues, J. J. P. C. (2022). Adaptive Intrusion Detection in Edge Computing Using Cerebellar Model Articulation Controller and Spline Fit. *IEEE Transactions on Services Computing*, *16 (2023)*(2), 900-912. https://doi.org/10.1109/TSC.2022.3178471

Adaptive Intrusion Detection in Edge Computing Using Cerebellar Model Articulation Controller and Spline Fit

Gulshan Kumar, *Member, IEEE*, Rahul Saha, Mauro Conti, *Fellow, IEEE*, Reji Thomas, Tannishtha Devgun, and Joel J. P C. Rodrigues, *Fellow, IEEE*

Abstract—Internet-of-Thing (IoT) faces various security attacks. Different solutions exist to mitigate the intrusion problems. However, the existing solutions lack behind in dealing with heterogeneity of attack sources and features. The future anticipated demand of devices' connections also urge the need of new solutions addressing the concerns of time consumption and complexity. In this article, we show a novel solution for the intrusion detection in IoT framework. We configure the intrusion detection in the edge computing layer so that the effect of the attack is not propagated to the clouds. Our solution uses cerebellar model articulation controller with kernel map. This combination is very new in the direction of intrusion detection; hence, it emphasizes the novelty of our proposed intrusion detection solution. We name our solution as *Cerebellar Model Articulation Controller based Intrusion Detection System (CMACIDS)*. Additionally, we use spline fitting to the kernel mapping for the model fit; this adds on another novel contribution to CMACIDS. The results obtained with our detection system are compared with the state-of-the-art solutions in terms of complexity, false alarms, and precision of detection. The analysis of the comparative study proves the efficiency of the solution and makes CMACIDS suitable for IoT paradigm.

Index Terms—Intrusion, IoT, edge, learning, artificial intelligence, kernel, cloud, spline

1 Introduction

The present technology evolution centers around Internet-of-Things (IoTs). Healthcare, transportation, supply chain management, manufacturing industries are some of the dominant application areas of IoTs. In such applications, all the connected devices communicate with the user-driven commands from mobile devices [1], [2]. Technology predictions give the hint of increasing number of connections by each year; the commencement of 5 G and beyond significantly contributes to the exponential growth of IoTs [3]. The manufacturing industry, e-healthcare, smart home, smart city, smart cars, supply chain management, etc., use IoT infrastructure [4], [5], [6]. Among various proposed architectures and frameworks, two-tier and three-tier architectures of IoTs are more popular [7]. A basic 3-layered architecture of IoT infrastructure consists of perception layer, fog layer, and cloud. We show a schematic diagram of the connection of these layers in Fig. 1. Perception layer and fog layer combined known as edge in IoTs.

Basically, IoT architecture and/or framework has two major layers: perception and communication. In the perception layer, the devices connect to the IoT and communicate with other devices. The intermediate layer(s), such as fog layer (depending on architecture), helps in reducing the delay of the communication between the cloud and the perception layer. The huge number of device to device communication, increases protocol varsity, large attack surface, and the increased sophistication of attack vectors compromise the benefits of IoTs in terms of security. The predicted number of device connections over 20 billion increases the data volume in zettabytes and more; similarly, the large attack surface also increases the possibility of zero-day vulnerabilities. This is enough to create a massacre of data stored in the cloud if managed casually and security is not assured. However, security and privacy have always been concerns in networking. With IoT, these issues increase manifold as the vulnerability space is larger than the conventional networking technology [8], [9]. Various attacks, both passive and active, on IoT applications need special security

- *Gulshan Kumar and Rahul Saha are with the School of Computer Science, Engineering, Lovely Professional University, Jalandhar 144001, India, and also with Department of Mathematics, University of Padua, 35122 Padova, Italy. E-mail: {gulshan3971, rsahaaot}@gmail.com.*
- *Mauro Conti is with the Department of Mathematics, University of Padua, 35122 Padova, Italy, and also with the Delft University of Technology, 2628 Delft, CD, Netherlands. E-mail: conti@math.unipd.it.*
- *Reji Thomas is with the Division of Research, Development, Lovely Professional University, Jalandhar 144001, India. E-mail: rthomas.eyyalil@gmail.com.*
- *Tannishtha Devgun is with the Department of Mathematics, University of Padua, 35122 Padova, Italy, and also with Nokia Solutions, Networks India Pvt. Ltd, Karnal, Haryana 132001, India. E-mail: jmd.tannishtha@gmail.com.*
- *Joel J. P C. Rodrigues is with the College of Computer Science, Technology, China University of Petroleum (East China), Qingdao 266555, China, and also with Intituto de Telecomunicações, 6201-001 Covilhã, Portugal. E-mail: joeljr@ieee.org.*

Manuscript received 13 Apr. 2021; revised 4 Apr. 2022; accepted 25 May 2022. Date of publication 27 May 2022; date of current version 10 Apr. 2023.
This work was supported in part by the European Union's Horizon 2020 Research and Innovation Program for the project COLLABS Under Grant 871518, in part by FCT/MCTES through national funds and when applicable co-funded EU funds under Grant UIDB/50008/2020, and in part by Brazilian the National Council for Scientific and Technological Development - CNPq, via under Grant 313036/2020-9.
(Corresponding author: Rahul Saha.)
Digital Object Identifier no. 10.1109/TSC.2022.3178471

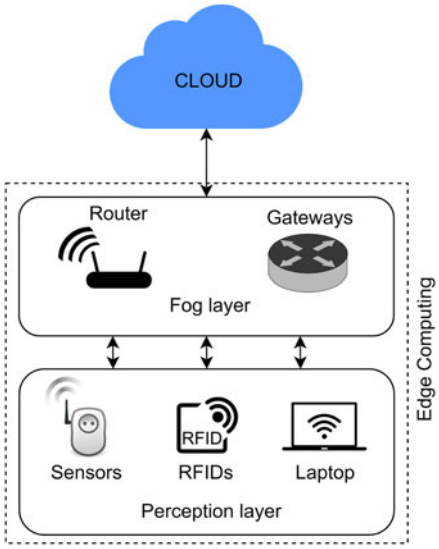


Fig. 1. Generic IoT framework.

protocols and algorithms. It is well known that by enhancing the functionalities of the security approaches, intrusion detection comes into existence and has been considered as the first step of defence in any security framework [10]. Various intrusion detection approaches have been introduced in recent times for wireless networks and some of them are applicable to IoT as well. However, their performance are in question due to their convergence time, increased complexity, lack of heterogeneity assumptions, and high resource consumption; thus, those approaches need rigorous update or extension. Therefore, it motivates us to research for an intrusion detection framework in IoT that provides low latency, less energy consumption, and less complex. Furthermore, as the devices join the network in the perception layer of IoT framework, it is better to include an intrusion detection mechanism in the edge computing layer, composed of both perception and fog. As the devices in the perception layer are resource-constrained, the developed intrusion detection must be less complex to make it suitable for resource-constrained environments. Therefore, in our present work, we develop an intrusion detection model for IoT using Cerebellar Model Articulation Controller (CMAC) and Spline Fit. This approach makes the detection process automated and learning-based.

1.1 Motivation and Contribution

The increasing demand of devices and communication pose serious insecurity probabilities on IoT infrastructure. Even though various security solutions exist, the sophistication of the attackers' tools make it necessary to update the solutions. Therefore, in our proposed approach, Cerebellar Model Articulation Controller based Intrusion Detection System (CMACIDS), we follow a new strategy of learning with cerebellar model and also use b-spline fit for system stability which is never explored before in the existing literature. We use reinforcement learning with cerebellar model in the control actions based on activated weights. Reinforcement learning provides the benefits of i) adaptability, ii) reinforcement learning does not require large labeled datasets, iii) bias resistance, iv) goal-oriented and consideration of sequence of tasks, and v) long-term sustainability due to reinforcement. The benefits of reinforcement learning and CMAC urge to experiment our work with CMAC and b-spline fit. The main contribution of the present work is as follows:

- We address the problem of intrusion detection in IoT framework and provide a solution to the problems of multidimensional enablers of perception. The solution is configurable in edge computing layer leveraging the perception layer complexities.
- The combination of cerebellar model articulation controller for automated learning and classification, kernel maps for feature extraction and vector generation, and spline curve fitting for fast stabilization of the system provides a novel and beneficial feature for CMACIDS. The model articulates based on the weight vectors and each stage it provides a solution converging faster towards the global optima. Thus it names after model articulation controller.
- The enablers help to create the feature extraction process multi-dimensional leading to more accurate detection and less false alarms. Cerebellar model articulation controller uses reinforcement learning and provides the advantages of linear memory usage, pipelined hardware, least square error without learning rate tuning. The experimented analysis and comparative study based on various type of attack detection, latency and complexity confirm the superiority of the presented work.

The practitioners (academia and industry) can use the proposed cerebellar model on the servers of a Security Operation Centre (SoC). Besides, being the less complex and low resource consumption method, cerebellar articulation with spline fit provides and efficient framework for IoT intrusion detection. Therefore, any IoT framework aiming for security operations can include our proposed approach with the benefits of ease of convergence to the optimum detection capability, pipelined hardware, linear memory usage, and auto-learning feasibilities.

1.2 Organization

The rest of the paper is organized as follows. Section 2 reviews the recent times development in the direction of intrusion detection. Section 3 explains the technical features and functionalities of the proposed approach. Section 4 shows the results. Section 5 concludes the paper.

2 Related Work

In this section, we review some of the recent works in the direction of intrusion detection in IoTs. We segregate the approaches into two parts. The first part considers intrusion detection models available for IoT-based applications and the second part discusses some general intrusion detection approaches for Wireless Sensor Networks (WSNs) as WSNs are the integral parts of the IoT infrastructure.

Initially, some machine learning-based approaches have been identified for intrusion detection in IoT [11], [12].

Among them, Restricted Boltzman Machines for intrusion detection has been proved significant due to its unsupervised learning methodology and high feature extraction ability from the data generated by sensor infrastructure [13]. In this method, different classifiers are used for training on the basis of extracted features which provide detection accuracy. However, the feature loss problem with a high number of feature extraction is a concern in this method. Following the same line, deep migration learning is also used for feature extraction and implemented in the intrusion detection for smart cities' [14]. Though this method claims to be efficient but, it is not well suited for edge computing. Wormhole attack prevention for IoT based Routing protocol is considered for low power and lossy network [15]. Received Signal Strength Indicator (RSSI) used for intruder detection may lead to camouflage attacks. Adversarial reinforcement learning method has also been used for the intruder detection that includes the environment's behavior for the learning process [16]. The main drawback of this iterative two-staged agent-based learning process is the complexity to tolerate minimum latency. Moreover, the dynamic changes in the environment may result in 'non-negotiable recursive adjustment' with null. The 'fuzzy' and 'fast fuzzy' pattern tree techniques for maliciousness detection are other significant contributions in IoT security [17]. The use of fast fuzzy patterns gives good accuracy value of the detection. However, with the huge amount of heterogeneous data this approach falls apart with the curse of dimensionality problem and increased computational complexity. This is one of the techniques used for the performance comparison in the present study. Game-theoretic detection of intrusion has also been experimented recently [18]. The approach uses the incentive mechanism for intrusion detection to get the optimal solution. Further, a punishment-appeal step is also induced to avoid compromising behavior among the nodes. Though the approach is novel, it is unable to prove its strength in the environment of IoTs as the number of nodes (devices and connections) is large and optimal solution for the incentive is complex with dynamic situations. Energy-efficient and quick detection of intrusion is well experimented based on Gaussian distribution [19]. The method emphasizes on the connections among nodes and establishes a relationship among the network parameters and the detection capability. However, being a very dynamic heterogeneous network, the prediction of deployment with Gaussian distribution is not the best suit for the real IoT environment.

Apart from the machine learning approaches, recently developed blockchain solutions for intrusion detection in IoTs are noteworthy [20]. One such approach uses blockchained signature-based intrusion detection system that updates the trusted signature database incrementally and collaboratively. It removes the indulgence of trusted intermediaries for verification by utilizing the distributed architecture of blockchain. The process claims to be efficient but with the increasing number of device connections, the collaborative consensus time increases which may lead to mismanaged intrusion alert generation and detection. As a part of IoT, automated systems also require an intrusion detection mechanism. An approach for Building Automation System (BAS) uses a 'context-aware data structure' by modeling the heterogeneous information acquired from building assets [21]. It then analyzes the data structures with anomaly-based detection mechanism with suitable learning methods. Hierarchical data management and its intrusion detection with 'low false-negative' is a necessity in real-life implementations. Especially, the anomaly-based intrusion detection techniques are more prone to have this problem.

Learning-based hierarchical intrusion detection with sub-classes is an efficient solution to address this problem [22]. However, the approach does not confirm the contextual segregation of hierarchical data which may lead to privacy breach in later stages. Artificial neural network-based approach for intrusion detection is worth mentioning here [23]. This approach is basically used for deep packet inspection; the authors have implemented the process for offline shell-code detection only. IoT deals with online data in huge amount and therefore, the approach is not appropriate for the IoT infrastructure. It requires an extension for IoT applicability. Researchers also use semi-supervised learning approach for generating intrusion detection framework for IoTs [24]. The learning process is configurable in the fog layer with Extreme Learning Machine (ELM) and Fuzzy C-Means to obtain faster detection rate. This approach is distributed in nature and enables attack detection and networks edge. The process claims to be efficient in lowering the detection time and with moderate-to-high detection accuracy. This approach along with fuzzy' and 'fast fuzzy' pattern tree techniques has been chosen for the performance comparison in the present study.

Apart from the above IoT-based solutions for intrusion detection, some generic solutions for WSNs, Mobile Ad hoc NETworks (MANETs) and mesh networks are worth mentioning here to know the current status of all the major intrusion detection models in the reecent years. Black hole attack is one of the major intrusions in MANETs. A fuzzy-based solution to address this problem is noteworthy [25]. Approaches are also developed to detect a Denial-of-Service (DoS) attack. Such an algorithm uses Radial Basis Functions (RBF) and Support Vector Machine (SVM) with AdaBoost to improve the performance of the detection and classification [26]. Another DoS detection-based solution has been investigated with low power consumption [27]. Multiple support vector machine and genetic algorithm-based feature selection are also under consideration by the researchers [28]. Though the authors have not confirmed its applicability in IoTs, we select this algorithm as one of the candidates for comparing the results obtained with the proposed approach. Jamming attack detection uses a Network Intrusion Detection System (NIDS) with Dempster-Shafer Theory of Evidence [29]. A game-theoretic approach has received attention due to its ability of prediction for attack patterns [30]. The model uses auto-regression and claims to balance detection efficiency and energy consumption. Signaling game has also been executed earlier [31]. Intrusion Detection System (IDS) functions are utilized in many ways to achieve a practical solution for intrusion detection specifically for mesh networks with energy constraints and collaborative attributes [32]. This approach aligns to the consideration of traffic agnostic and traffic-aware centralized and distributed algorithms [33]. Earlier, the cooperativeness of these approached

TABLE 1
List of Symbols and Notations

Description	Symbol/Notation	Equation no.
High dimensionality space	$\mathfrak{S}$	(1), (3), (4), (5)
Non-linear mapping	$\mathscr{T}$	(3), (4), (5), (6)
Dot product of two elements	$< . >$	(1), (2)
Kernel function	$\mathscr{K}$	(1),(2)
Within class scatter matrix	$\mathfrak{S}_W^{\mathscr{T}}$	(3)
Between class scatter matrix	$\mathfrak{S}_B^{\mathscr{T}}$	(4)
Learning decaying factor	$\mathscr{D}$	(11)
Learning rate	$\mathscr{L}$	(11),(12)
Learning sensitivity function	$m(\hat{o}_i^t)$	(11),(12),(13)
Cost function of switching	ς	(14)

has been emphasized thoroughly with multi-objective optimizing problems [34].

The above discussion shows that a number of research works are available in the literature to solve the problem of intrusion detection in IoT and non-IoT perspectives. Each of them has its pros and cons. With the increasing number of devices and the requirement of lower power consumption, higher accuracy, and less latency urge to develop a new solution, which results in our present solution of intrusion detection in IoTs as CMACIDS.

3 CMACIDS Approach

The proposed approach addresses the intrusion detection issues in the perception layer of IoT-Cloud interfacing networks. We define an adaptive learning approach using the Cerebellar Model Articulation Controller (CMAC) and also have used Spline interpolation for the best suit detection [35], [36]. We name our proposed model *"CMAC based Intrusion Detection with Spline (CMACIDS)"*. The selection of this method for our proposed work is majorly derived from the advantages of the method including less computational complexity, one step convergence, linear memory usage, applicability in a significant noisy network environment and automated classification process. To the best of our knowledge, CMACIDS is the first intrusion detection method to apply cerebellar model with b-spline fit and kernel maps obtaining certain advantages for IDS in IoTs. To realize our novel intrusion detection, CMACIDS approach gathers the network data by the intrusion analyzer installed in the edge level of IoT cloud interface.

3.1 System Model

We show our proposed model of CMACIDS in Fig. 2. We keep the basic three-layered functionalities of the generic IoT framework intact in the framework. However, we add an intrusion analyzer to the generic IoT framework. It is a combination of functionalities included in the fog layer. These functionalities, e.g., routing, switching, local storage, etc. can be included in any device in the fog layer as per the requirements. In the present work, heterogeneous devices associated with the 'edge' generate heterogeneous data. By heterogeneous data, we mean that the application scenarios of the devices. As in the perception layer various devices connect, their application areas may differ, and moreover, the types of data they are transmitting also differ. Therefore,

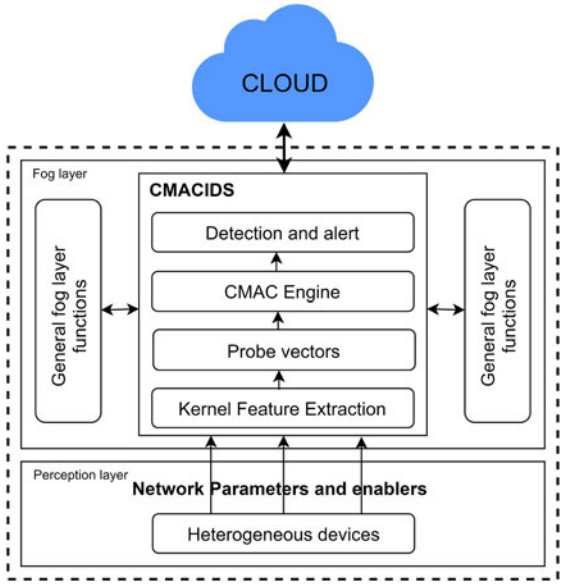


Fig. 2. CMACIDS system model.

we try to get the extracted features from this to have a suitable detection mechanism. These extracted features help in generating probe vectors. Probe vectors are given input in Cerebellar Model Articulation Controller Engine (CMAC Engine). We use spline interpolation for the better fit of the data classification. Once the classification is done properly, malicious detection verifies, and accordingly generates alerts depending on the sensitivity of the risk along with the necessary information. CMAC engine is responsible for model articulation and interpolation.

The Detection and Alert (DEA) performs the task of risk sensitivity calculation and updating the risk threshold values. Once the detection is done, the intended device or service is blocked. The system modeling of the proposed work has been classified in the following subsections: i) *Network feature extraction*, ii) *Probe vector generation*, iii) *Learning through Cerebellar Model Articulation Controller (CMAC)*, and iv) *Detection and update and spline fitting*.

We provide a list of notations and symbols along with their descriptions and equation reference(s) in Table 1.

3.2 Network Feature Extraction

The perception layer constituents are shown in Fig. 3. Three basic modules: Radio Frequency IDentification (RFID), Wireless Sensor Network (WSN) and RFID based Sensor Network (RSN) are the basic enablers for Edge-to-Cloud computing networks [37], [38], [39]. Referring to Fig. 3, we observe that the perception layer consists of various devices, sensors and RFIDs. Each of these nodes having their different data accumulation process. Therefore, CMACIDS collects these variable data and performs the kernel feature extraction for generating probe vectors. As a result, CMACIDS is able to handle the stochastic nature of input from the network traffic.

All these enablers are responsible for the data acquisition and transmission and the heterogeneity of the data is the outcome. Therefore, network probe vectors are essential for conceptualizing the network behavior and analysis for maliciousness. Network probe vectors are generated by Kernel Feature Extraction technique [40]. As the network data is

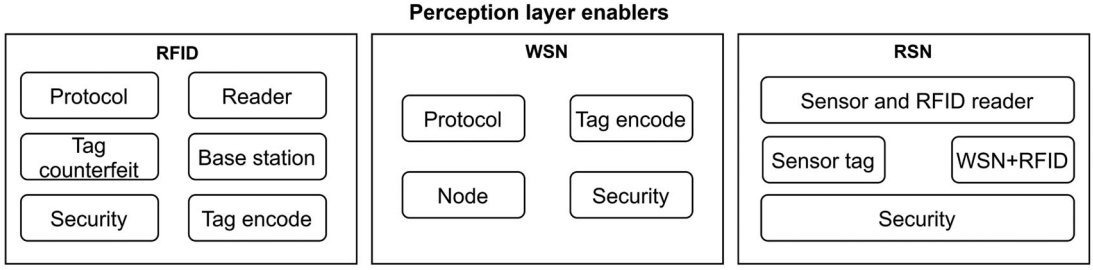


Fig. 3. Perception layer enablers.

enormous, kernel technique helps to extract the features with fast iterations and it additionally ensures precision. Also, kernel feature extraction method provides enough power of nonlinear discriminant for the classification of linearly non-separable classes. The basic idea of using this method is to transform the multidimensional network device data properties (data in transaction and device behavior) to a high dimensionality space, '$\mathfrak{S}$' via non-linear mapping, '$\mathscr{T}$'

$$\mathscr{T}R^d \rightarrow \mathfrak{S}, x \longmapsto \mathscr{T}(x),$$

where R^d is the multidimensional sample space. The mapped space is non-linearly related to the original space that contradicts the curse of dimensionality problem [41]. Therefore, we use kernel function to obtain $\mathscr{T}(x)$. Otherwise, linear classifiers, Radial Basis Network (RBN) or, boosting algorithms are also applicable for the mapping process; it is applicable if $\mathfrak{S}$ is not too high and less complex mapping is observed. Thus, the use of kernel functions in the present proposed approach helps to compute the dot products of the mapped samples in '$\mathfrak{S}$' such that the following equation holds.

$$\mathscr{K}(x,y) = (< x, y >) = < \mathscr{T}(x).\mathscr{T}(y) >, \tag{1}$$

where $< . >$ is the dot product. Similarly, for the polynomial distributions the above equations can be re-written as

$$\mathscr{K}(x,y) = (< x, y >)^n = < \mathscr{T}(x), \mathscr{T}(y) >^n, \tag{2}$$

where, n is the degree of the ordered products for x.

Let, $\{\mathscr{T}(x_1^1), \mathscr{T}(x_2^1), \ldots, \mathscr{T}(x_{N1}^1), \mathscr{T}(x_1^2), .., .\mathscr{T}(x_{NC}^c)\}$ represent the mapped samples in '$\mathfrak{S}$' space. Then within the class and between the class, scatter matrices are denoted as $\mathfrak{S}_W^{\mathscr{T}}$ and $\mathfrak{S}_B^{\mathscr{T}}$ and are given by

$$\begin{aligned}\mathfrak{S}_W^{\mathscr{T}} &= \sum_{i=1}^{C}\sum_{m=1}^{N_i} \mathscr{T}(x_m^i) - M_i^{\mathscr{T}})(\mathscr{T}(x_m^i) - M_i^{\mathscr{T}})^T, \\ &= (\mathscr{T} - \mathscr{T}\mathrm{G})(\mathscr{T} - \mathscr{T}\mathrm{G})^T, \end{aligned} \tag{3}$$

$$\mathfrak{S}_B^{\mathscr{T}} = \sum_{(i=1)}^{C} N_i(\mu_i^{\mathscr{T}} - \mu^T),$$

$$(\mu_i^{\mathscr{T}} - \mu^{\mathscr{T}})^{\mathscr{T}} = (\mathscr{T}U - \mathscr{T}L)(\mathscr{T}U - \mathscr{T}L)^{\mathscr{T}}, \tag{4}$$

where, $\mu^{\mathscr{T}}$ is the sample mean mapped data, μ_i^{ϕ} is the sample mean of the i_{th} class, and $\mathscr{T}$ is the mapping matrix whose columns are mapped with training set samples in $\mathfrak{S}$. $\mathrm{G} = diag[\mathrm{G}_1, \mathrm{G}_C] \in \mathfrak{K}^{(m\times m)}$ diagonal matrix and $\mathrm{G}_i \in \mathfrak{K}^{(N_i \times N_i)}$ is a matrix where all elements are $\frac{1}{N_i}$. $U = diag[u_1, u_2, \ldots, u_C] \in \mathfrak{K}^{(m\times C)}$, block-diagonal matrix and $u_i \in \mathfrak{K}^{(N_{i\times 1})}$ is a vector whose entries are all $\frac{1}{\sqrt{N_i}}$. $L = [l_1, l_2, \ldots, l_C] \in \mathfrak{K}^{(m\times C)}$, matrix where $l_i \in \mathfrak{K}^{(m\times 1)}$, a vector whose elements are $\sqrt{\frac{N_i}{m}}(N_i)$. The principal components are calculated by solving the eigenvalue problem as

$$\gamma w = \mathfrak{S}_{\mathscr{T}}^{\mathscr{T}} w, \;\; where \;\; \mathfrak{S}_T^{\mathscr{T}} w = \mathfrak{S}_W^{\mathscr{T}} + \mathfrak{S}_W^{\mathscr{T}}, \tag{5}$$

$$w = \sum_{(i=1)}^{C}\sum_{(m=1)}^{(N_i)} \alpha_{ij}(\Phi(x_m^i) - \mu^{\mathscr{T}}) = \mathscr{T} - \mathscr{T}l_m\alpha, \tag{6}$$

Substituting Equation (5) in Equation (6) we get

$$\gamma\bar{\mathscr{K}}\alpha = \bar{\mathscr{K}}^2\alpha \Rightarrow \gamma\alpha = \bar{\mathscr{K}}\alpha, \tag{7}$$

Where $\bar{\mathscr{K}} = \mathscr{K} - l_m\mathscr{K} - \mathscr{K}l_m + l_m\mathscr{K}l_m \in \mathscr{K}^{(m\times m)}$ and $\mathscr{K} \in \mathscr{K}^{(m\times m)}$ is given by as follows:

$$\begin{aligned}\mathscr{K} &= \mathscr{T}^{\mathscr{T}}\mathscr{T} \\ &= (\mathscr{K}_{mn}^{ij} = < \Phi(x_m^i), \Phi(x_n^j) \\ &> = \mathscr{K}(x_m^i, x_n^j))_{(i,j=1..,C;m=1..,N_i;n=1,..N_j),}\end{aligned}$$

There are at most $m-1$ eigen vectors corresponding to non-zero eigen values of $\bar{\mathscr{K}}$. Since $w_1, w_2, \ldots, w_{(m-1)}$ must be orthonormal set, the vectors α_j must be normalized in a way such that the above condition holds true. Then, we select the most significant n eigen vectors for the aforementioned 'feature extraction'. In our process, the extracted features with higher eigen values are: protocol, IP address, time, packet size, hop counts, command memory count, command write function, response memory, control code. In Table 2, we show the eigen values for which we have selected these parameters from the heterogeneous data. The table also shows the connection of these features with the

TABLE 2
Eigen Values for Selected Features

Feature name	Eigen value	Attack relation
Protocol	0.687	IP sweep, port sweep
IP address	0.876	IP sweep, port sweep
Time	0.766	IP sweep, port sweep
Packet size	0.688	SPY attack
Hop counts	0.692	Multihop
Command memory count	0.881	Buffer overflow, Rootkits
Command write function	0.632	Buffer overflow, Rootkits
Response memory	0.733	Buffer overflow
Control code	0.657	Rootkits

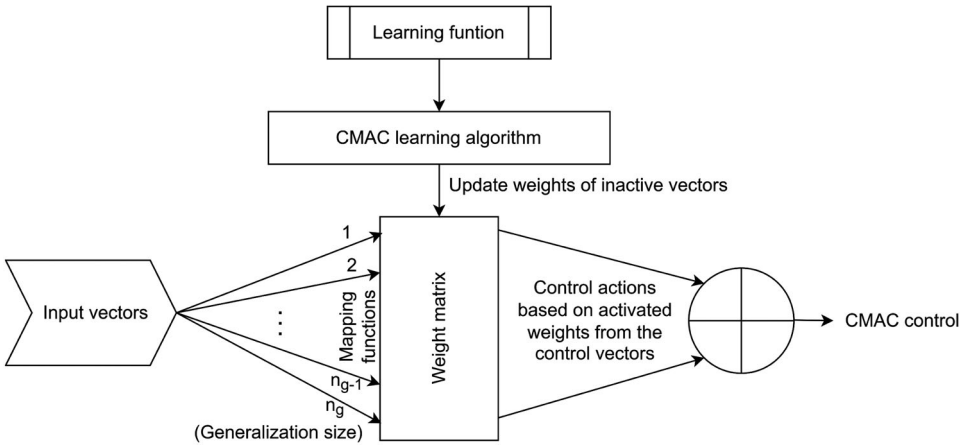


Fig. 4. Logical representation of CMAC processing.

attacks under consideration. We provide the attack description is given in Section 4.

3.3 Probe Vector Generator

Once the feature vectors are generated, all the selected features create a feature space F_s in F. All the samples x_i are transformed in this subspace with dot product projection using empirical kernel map [42] as

$$\mathscr{Z}_i = \mathscr{T}^t_{(S)}\Phi_i, \tag{9}$$

where $\Phi_i : x_i \rightarrow \mathscr{Z}_i$ and is given by $(\mathscr{K}(x_i,\ s_{i1}), \mathscr{K}(x_i, s_{i2}), \ldots, \mathscr{K}(x_i, s_{im}))$. The empirical kernel map is beneficial in the adaptation of any approach. It is less complex as compared to the other transformations such as orthogonal maps. It also provides good approximation with the kernalized vectors. Therefore, we choose this method for the data probing vectors. Further, to estimate the network probe vectors we apply linear regression. Given a dataset in the form of $(\mathscr{Z}_i, o_i)$ where $\mathscr{Z}_i$ is the kernel map projection onto F_s in F of samples x_i, the linear regression reduces the Mean Square Error (MSE) of the learning set L and outputs the estimated vector o_i which is then used with Moore-Penrose pseudo inverse [43] as follows.

$$\hat{o}^t_i = \mathscr{Z}^t_i A + \beta^t, \tag{10}$$

where, $A = (Z^t_T Z_T)^{(-1)} Z^t_T O_T$ *and* $Z_T = (\mathscr{Z}^t_i)_{(i\in T)}, O_T = (o^t_i)_{(i\in T)}$. The vector β is added to the estimated 'vector A' which is formed by adding a constant component in each vector $\mathscr{Z}^t_i$. All the $\hat{o}^t_i$ are then given as inputs for Cerebellar Model Articulation Controller (CMAC) for learning and classification of intrusion detection.

3.4 Learning Through Cerebellar Model Articulation Controller (CMAC)

We use Cerebellar Model Articulation Controller (CMAC) for the classification of the degree of intrusion. This model provides automated classification with reinforcement learning and shows recursively least square error without tuning of learning rate. Moreover, the computational complexity is $O(N)$ and can be made to converge in one step with linear memory usage and pipelined hardware. The parametric memory cells of 4 MB have been used for this purpose, where weights and required features are changed continuously with iterations for minimizing the error. Note that this memory requirement in the cumulative requirement of CMACIDS process which is quite low. Thus, it satisfies the memory utilization criteria for the applicability in resource constrained IoTs.

We use a gradient type learning rule for updating the weights [44]. The continuous change of leaning rate affects the CMAC performance and we observe that the best value of the initial learning rate $\mathscr{L}_i$ has to be unity. The variable learning rate equation is derived as

$$\mathscr{L}(\hat{o}^t_i) = \mathscr{L}_i(1 - \mathscr{D}m(\hat{o}^t_i) \sum_{(j=0)}^{k} |e_j|), \tag{11}$$

where, $\mathscr{D}$ is the learning decaying factor, $m(\hat{o}^t_i)$ is the learning sensitivity function which is a nonlinear learning switching function defined as

$$m(\hat{o}^t_i) = \begin{cases} 0, & (|e(k)|_{mean} \geq \varepsilon)\ \ AND\ \ \mathscr{L}((k-1)=0) \\ 1, & \text{otherwise} \end{cases}, \tag{12}$$

where ε is the permissible absolute mean error. Control affine switching function for the experimentation is defined as

$$m(\hat{o}^t_i)_t = F_j(v(\hat{o}^t_i)) + G_j(\hat{o}^t_i)u(\hat{o}^t_i), \tag{13}$$

Where, $F_j : R^n \rightarrow R^n$ and $G_j : R^n \rightarrow R^(n \times m)$, $\forall j \in \tau \equiv 1, 2, \ldots, M$, n is the dimension of the state vector $v(\hat{o}^t_i)$ and m is the dimension of control vector $u(\hat{o}^t_i)$ The switching sequence considered as: $\mathfrak{S} = ((t_0, j_0), (t_1, j_1), \ldots.., (t_k, j_k))$; $t_0 \leq t_1 \ \leq \ldots \leq t_k \leq t_f; 0 \leq k \leq \infty$and $j_k \in \tau$ The sequence parameter (t_k, j_k) means that the system switches from $j_{(k-1)}$ to j_k at time t_k and k denotes the number of switching. The cost function of the switching has been considered and mathematically derived as follows:

$$\varsigma = \Psi(v(\hat{o}^t_i)) + \frac{1}{2}\int_{(t_0)}^{(t_f)} (Q(v(\hat{o}^t_i)) + u(\hat{o}^t_i)^\tau R u(\hat{o}^t_i))dt, \tag{14}$$

Q and Ψ are convex smooth positive semi definite functions given as: $R^n \rightarrow R.\Psi$ penalize the state vector control effort and R penalizes the control effort. We show a schematic diagram of CMAC processing in Fig. 4.

We train CMAC with the cost function composed of the structural response and the control affine function as shown in Equation (14). We use gradient descent rule applied to the cost function at the kth step to obtain an optimal solution. The weight update at the kth step is given by

$$\Delta W_I^l = -\eta \, \frac{\delta \varsigma_k}{\delta W_I^l}. \tag{15}$$

In Equation (15), η is the training rate and W_I^l are the invoked weights. To avoid the the over-fitting issues, we observe that the training rate must be less than equal to the learning rate.

3.5 Detection and Update

The functionalities of the CMAC are further enhanced by using the Kernel-Based Online Anomaly Detection (KBOAD) technique as it is well suit for the online applications with kernel mapping [45]. Also, multivariate traffic and kernel function $\mathscr{Z}_i = \mathscr{T}_{(S)^t}\Phi_i$ have been used to generate the probe vectors in the feature space F_s. These vectors $\hat{o}_i^t$ generate an equivalent dictionary $\mathscr{D}$. The feature error E from the project of $\hat{o}_i^t$ to the feature space F_s are calculated. We assume the upper and lower threshold to be ϑ_u and ϑ_l. CMACIDS considers two types of alarms for detection: alarm of severity and alarm of sensitivity. The former one is more severe when the observed feature vectors totally deviate from the dictionary. The latter one is considerably less severe. The change in feature space of network measured parameters may be the reason for this; these alerts are to be investigated further. The detection process executes as follows:

If $(E < \vartheta_l)$, $\hat{o}_i^t$ is linear dependent on $\mathscr{D}$,

Else if $(E > \vartheta_u)$, $\hat{o}_i^t$ is deviated from the normal behaviour and $\hat{o}_i^t \mathscr{D}$, generate the alarm of severity

else if $(\vartheta_l > E > \vartheta_u)$, $\hat{o}_i^t$ is linearly dependent on $\mathscr{D}$, generate the alarm of sensitivity

We further investigate the alarm of sensitivity to evaluate the usefulness in terms of resolution. The usefulness of an alarm generating feature vector $\hat{o}_i^t$ is assessed by observing the kernel map values with respect to time $t = 1, 2, \ldots t + l$. If a kernel value is high, then $\mathscr{Z}_i$ is higher and such significant-high kernel values tell that the $\hat{o}_i^t$ is not malicious, it is only shifted to a new feature space after a time interval $\Delta t = t_i + L_C$, where L_C is the cumulative lag and $l \in L_C$. On the other hand, if significant kernel values are small then $\mathscr{Z}_i$ is also small and $\hat{o}_i^t$ is detected as an anomaly. As the network parameters are heterogeneous and a huge number of devices are connected at the perception layer, it is obvious that after a time interval, the parameters change and as a result, the feature spaces are also going to be changed which has been also considered in the solution. After $t + l_C$, the changes in the model lead to either of the three functions: i) update dictionary, ii) change of alarm from severity to sensitivity, and iii) change of alarm from sensitivity to severity. For this, an update condition is evaluated and compared as following.

$$\left[\sum_{(i=t+1)}^{(t+l_C)} \approx(\mathscr{K}(x,y) \; > \; d)\right] \; > \; \gamma l_C, \tag{16}$$

where $\approx$ is the indicator function, d is the kernel value threshold, γ is constant $\in (0, 1)$. If the condition is found true, then the sensitivity of the alarm is changed to normal behavior. However, if it is false then the alarm of sensitivity is changed to the alarm of severity. l_C is kept relatively small as the time lag helps for precise decision.

3.6 Spline Fitting

To test the proposed approach, we apply the concept of B-spline fitting [46]. Probe vectors are converted to the polynomials of degree $n-1$. The spline is made up of degree n. Here, the feature mapped vectors are considered to be the knots in the splines. With a given sequence of knots, up to a scaling factor of s with respect to time lag l_C, the spline function $\mathfrak{B}(\mathscr{K})$ must satisfy the following condition.

$$\mathfrak{B}_{(s,l)}(\mathscr{K}) = \begin{cases} 0, (if \; \mathscr{T}^T \mathscr{T}.\mathscr{T}'^T \mathscr{T} = \; I \; where \; \mathscr{T}' \; is \; the \\ kernel \; output \; after \; l \; lag \; time \; and \; I \; is \\ the \;\; identity \; matrix) \\ non - zero, \; otherwise \end{cases}. \tag{17}$$

If the kernel function produces the same kernel eigen factors even after l_C lag period, it means that the spline knots are stable, and CMAC can be performed. Detection process becomes fast. So, the spline fit enhances the stability of the proposed model. We summarize the overall process of the proposed work in Algorithm 1.

Algorithm 1. CMACIDS Approach

1: Input: Perception Layer Enablers, multivariate traffic $\mathfrak{I}$
2: Output: Alert for Intrusion detection
3: Acquire perception enablers (network parameters)
4: Apply Kernel Feature Extraction technique $TR^d \rightarrow \mathfrak{S}, x \mapsto T(x)$
5: Define $\mathscr{K}(x,y) = < \mathscr{T}(x).\mathscr{T}(y) >$,
6: Calculate $\mathfrak{S}_W^{\mathscr{T}} = \sum_{(i=1)}^{C} \sum_{(m=1)}^{(N_i)} (\mathscr{T}(x_m^i) - \mu_i^{\mathscr{T}})(\mathscr{T}(x_m^i) - \mu_i^{\mathscr{T}})^T = (\mathscr{T} - \mathscr{T}G)(\mathscr{T} - \mathscr{T}G)^T$
7: Calculate $\mathfrak{S}_B^{\mathscr{T}} = \sum_{(i=1)}^{C} N_i(\mu_i^{\mathscr{T}} - \mu^{\mathscr{T}})(\mu_i^{\mathscr{T}} - \mu^{\mathscr{T}})^T = (\mathscr{T}U - \mathscr{T}L)(\mathscr{T}U - \mathscr{T}L)^T$
8: $\gamma w = \mathfrak{S}_T^{\mathscr{T}} w$, where $\mathfrak{S}_T^{\mathscr{T}} w = \mathfrak{S}_W^{\mathscr{T}} + \mathfrak{S}_B^{\mathscr{T}}$
9: $\mathscr{K} = \mathscr{T}^T \mathscr{T} = (\mathscr{K}_{mn}^{ij} = < \Phi(x_m^i), \Phi(x_n^j) > = \mathscr{K}(x_m^i, x_n^j))_{(i,j=1..,C;m=1..,N_i;n=1,..N_j)}$
10: Select the most significant n eigen vectors for feature extraction and goto Step 3.
11: Apply the kernel map with $z_i = \mathscr{T}(\mathscr{S})^t \hat{o}_i$
12: $\hat{o}_i^t = z_i^t A + \beta^t$
13: Apply CMAC
14: Initialize $m(\hat{o}_i^t)_t = F_j(v(\hat{o}_i^t)) + G_j(\hat{o}_i^t)u(\hat{o}_i^t)$
15: Derive learning rate as: $\mathscr{L}(\hat{o}_i^t) = \mathscr{L}_i(1 - Dm(\hat{o}_i^t) \sum_{(j=0)}^{k} |e_j|)$
16: $\varsigma = \Psi(v(\hat{o}_i^t)) + \frac{1}{2}\int_{(t_0)}^{(t_f)} (Q(v(\hat{o}_i^t)) + u(\hat{o}_i^t)^\tau R u(\hat{o}_i^t))dt$
17: Set the upper and lower threshold to be ϑ_u and $\vartheta_l$and calculate E on the projection of $\hat{o}_i^t$ to F
if $(E < \vartheta_l), \hat{o}_i^t$ is linear dependent on $\mathfrak{D}$,
else if $(\vartheta_l > E > \hat{o}_u), \hat{o}_i^t$ is linearly dependent on $\mathfrak{D}$, generate the alarm of sensitivity
else, $\hat{o}_i^t$ is deviated from the normal behaviour and $\hat{o}_i^t$ $\mathfrak{D}$,generate the alarm of severity
18: Update if : $[\sum_{(i=t+1)}^{(t+l)} \approx(\mathscr{K}(x,y) \; > \; d)] \; > \; \gamma l$
19: Apply Spline fit for stability

TABLE 3
Technical Specification of Devices

Mobile phones (5 nos)	Samsung Note 5 (2), Nokia Note pro (1), Redmi Note 7 pro (2)
Laptops (5 nos)	Dell Intel Core i7 (7th generation) (5)
Sensors (20 nos)	Atmel ATmega328 (5), ARM 32 b Cortex M3STM32F103 (5),Atmel ATmega128 L (5), Jennic JN5148 (5)
Switch (2 nos)	Cisco SG350-28-K9-EU 28-port Gigabit Managed Switch
Router (2 nos)	Cisco ISR4221/K9 Router
Gateway (1 nos)	Cisco ASR 1000

4 Performance Evaluation

In this section we discuss about the experimental procedure, results and their comparative analysis.

4.1 Experimental Setup

The experimental setup of IoT infrastructure developed for the present work consists of 30 end devices including 5 mobile phones, 5 laptops, and 20 sensors in the perception layer; fog devices include two switches, two routers, and one gateway. The experiments are performed in the Campus Area Network (CAN) with an average data transfer speed of 100 Mbps. The technical specifications of the devices used are shown in Table 3. We consider the attacks described in IoT attack ecosystem as in [24]: Port sweep, IP sweep, SPY, IMAP, Buffer overflow, rootkit, and Multihop are the attack types considered for the present study. For authentication attacks, basic authentication vectors available from the network traffic are only used. To detect the extended authentication attacks, a dictionary needs to be made for identifying the authentication vectors which has been kept for future work. The definitions of the attacks and their generation in the our experiments are shown in Table 4.

Synthesized Dataset. Our synthesized dataset includes one lakh of records. we consider seven attacks as in Table 4. 50% of the data is normal traffic and another 50% of the data is malicious traffic with some attacks. Sweep attacks are 10%, SPY attack is 5%, IMAP attack is 5%, buffer overflow attack is 15%, rootkit is 10%, multihop attack is 5%.

Data Pre-Processing. In our synthesized dataset, we have not used any pre-processing of data as the data is generated by our own deployed network. In the available dataset such as UNSW, NSL-KDD data pre-processing can be used to avoid incomplete features or non-required fields.

4.2 Performance Metrics

The performance of the proposed system is evaluated and then compared with other existing systems. The technical definitions are; i) True Positive (TP) - indicates that a malware is correctly identified, ii) True Negative (TN)-indicates that a benign is detected as a non-malicious application correctly, iii) False Positive (FP) - indicates that a benign is falsely detected as a malicious application, iv) False Negative (FN)- indicates that a malware is not detected and labelled as a non-malicious application and these definitions can be seen elsewhere [17]. By following the aforementioned definitions, parameters for the verification of the correctness of the systems are maintained viz., accuracy, precision, and recall. Accuracy defines the fact of the correctness of the system with the total number of truthful detection divided by all the detection.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}. \tag{18}$$

TABLE 4
Attacks Definitions

Attack type	Attack definition	Generation of attack
IP Sweep	An address sweep occurs when one source IP address sends a predefined number of ICMP packets to various hosts within a predefined interval of time.	We create a separate system as an attacker; it sends ICMP echo packets to the nodes in the configuration
Port Sweep	A port scan is an attack that sends client requests to a range of server port addresses on a host to obtain an active port and exploiting a vulnerability of that service.	We send general service requests to all the ports, the responses from the ports give the active ports and further port-based attack is created.
SPY attack	Transmits spyware (malware) with normal traffic flow.	We use a port-based service on the open port of 443 with a spyware to detect the hashes and logins in a system
IMAP attack	Malicious access to IMAP Server may use password spraying, DNS listing, etc.	We disguise with an permissible IP address and try to ping to the IP address of IMAP server
Buffer overflow	Transmitting data more than the capacity or predefined threshold of traffic flow so that data overflows into adjacent memory spaces and thus corrupting data, generally executed by transmitting shellcode.	We use malfunctioned pings to create this attack with extra parameters in the operation field of the payload
Rootkit	Rootkits work as malware usually transmitted with network traffic so that once it is installed, it can bypass the access privileges or gain the administrator control.	We create a rootkit and put in inside the data packet
Multihop	Disguises or camouflages the source of malicious traffic as multiple adversaries chain together as multiple proxies; identification of the original source is hard.	We use the tor command and control, a multicast address of attackers from the environment to launch this attack with flood

TABLE 5
Comparison of Correctness

	Accuracy	Precision	Recall
Dovom *et al.*, [17]	98.6%	98.8%	98.87%
Daming *et al.* [14]	93%	89.9%	95%
Rathore *et al.* [24]	90.33%	91.23%	92.55%
Proposed CMACIDS	99.37%	100%	100%

The values are calculated for all the instances and mentioned here in average only.

Precision defines the fact of the correctness of the system with the total number of intrusions correctly detected.

$$Precision = \frac{TP}{(TP + FP)}. \tag{19}$$

Recall defines the fact of the correctness of the system with the ratio of malware samples that are correctly predicted as per the total number of samples.

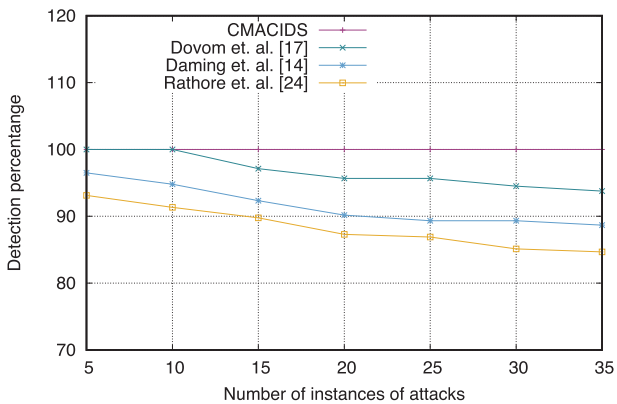
$$Recall = \frac{TP}{(TP + FN)}. \tag{20}$$

The three existing algorithms are chosen for performance comparison of the present model [14], [17], [24]. The reason for choosing these three existing approaches are due to their similar features of learning mechanisms, dealing with heterogeneous data. In the comparative analysis, we only adopt the methodology from the literature and apply them on our synthesized data to evaluate the performance. For our synthesized data, we follow the format of NSL-KDD dataset; it considers the following attributes: source and destination IP address, source and destination ports, CPU usage, memory usage, access controls, buffer size, intermediate hops, service requests and flags for TCP connections, number of ECHO packets, inbound and outbound traffic size, timestamps, and format of tor command.

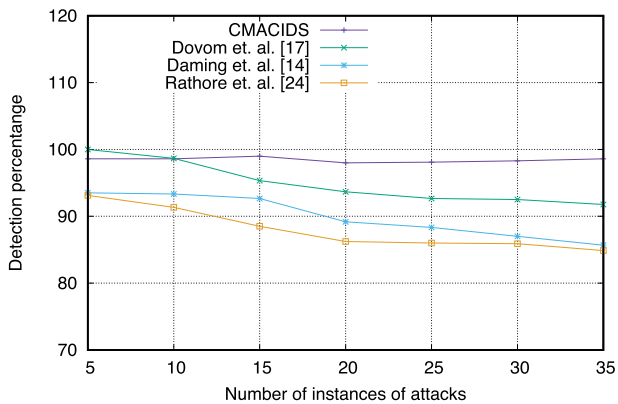
4.3 Results and Discussion

The comparison of the mentioned attributes is shown in Table 5. Table 5 shows that the proposed model is better in accuracy and achieves 100% of performance in precision and recalling the event. The reinforcement learning in the system and suitable method of kernel feature extraction have assisted for this advantageous performance. We also measure the candidate models of [14], [17], and [24] with the same network model and system configuration.

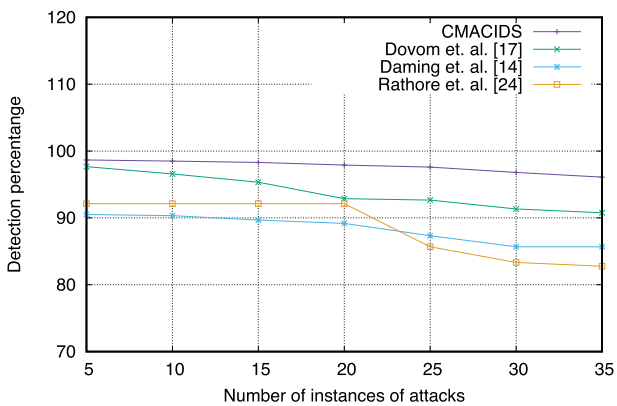
Further, attacks in increments of variations have been created to extend the above results and the behavior of the algorithms have been analyzed. The accuracy parameter is used for comparing the behavior with respect to increased variations and number of attacks and is shown in Figs. 5a, 5b, 5c, and 5d. In this study, four attack scenarios are considered; i) Scenario 1 considers sweep attacks including Port sweep and IP sweep and for that 50% Port and 50% IP sweep attacks are executed with the number of attacks



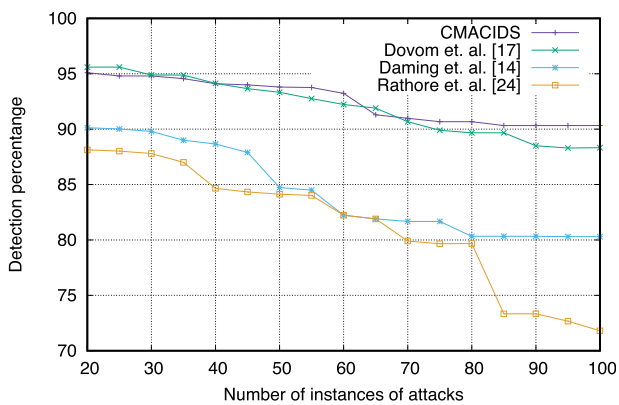
(a) Attack detection comparison (scenario 1)



(b) Attack detection comparison (scenario 2)



(c) Attack detection comparison (scenario 3)



(d) Attack detection comparison (scenario 4)

Fig. 5. Attack detection comparison in various scenarios.

TABLE 6
Single Attack Detection Time

Attack type	Dovom *et al.* [17]	Daming *et al.* [14]	Rathore *et al.* [24]	CMACIDS
Ping Sweep	15.89	17.02	12.07	13.38
IP Sweep	15.33	16.89	13.01	13.77
SPY	15.89	17.32	18.04	14.76
IMAP	18.26	15.33	19.57	14.02
Buffer overflow	18.33	19.37	28.07	17.33
Rootkit	10.67	10.33	25.67	16.40
Multihop	13.33	12.67	23.67	10.67
Average Communication Latency	74.67	98.19	108.67	53.33

The values are given in milliseconds.

varying from 5 to 35, ii) Scenario 2 executes the attacks of Scenario 1 and adds SPY and probing attacks, with number of attacks is varied from 5 to 35 with 1:2:3 ratio of the attacks, iii) Scenario 3 considers the attacks of Scenario 2 and has added buffer overflow and rootkit attacks, with number of attacks is varied from 5 to 35 with 30% scenario 1, 30% scenario 2 and remaining are of buffer overflows, and iv) Scenario 4 execute the attacks of Scenario 3 and adds Rootkit attacks and multi-hop attacks, with number of attacks is varied from 20 to 100.

As can be seen from Fig. 5a (scenario#1), proposed novel intrusion detection method, CMACIDS, performs with the accuracy of 100% in intrusion detection over the entire range of number of intrusion attacks, whereas [17] shows 100% accuracy up to 10 attacks and then falls down gradually with increase in the number of attacks. The other two algorithms [14] and [24] performs poor as in comparison. Scenario 1 confirms that CMACIDS is efficient enough in detecting sweep attacks accurately. However, in Fig. 5b (scenario#2), a change in CMACIDS behavior has been observed. It starts almost 97% accuracy and maintains the performance with the increasing number of attacks and its variations with a slight increase up to 97.8 % of accuracy and continues further with an average of 97.33% of accuracy. In comparison, the algorithm in [17] starts with an accuracy of 100 percent but at the 10 number of attacks, it coincides with the CMACIDS with its degraded performance and ends at approximately 92% accuracy at the end. CMACIDS average accuracy detection in this scenario is calculated as 97.8%. The other two algorithms [14] and [17] have experimented with 87.8% and 82.3% of average accuracy. The observation in scenario 2 signifies that even though Dovom *et al.* [17] starts detecting the variation of attacks with a high percentage but falls apart when the number of attacks increases and CMACIDS outperforms others in an average accuracy of detection with four variations of attacks. It can be seen from Fig. 5c (scenario 3) that CMACIDS is better as compared to all other algorithms in any variations of attacks. Rathore *et al.* [24] have shown promising efficiency at the starting point of scenario 3 but drastically degrades the performance after 20 number of attacks. Though all algorithms got affected by the variation, CMACIDS, and Dovom *et al.* [17] have shown a compatible performance of accurate detection with 92.3% and 89.67% in average over the entire range used for the present study. In scenario 4, the initialization of the experimentation has shown interesting behavior and is depicted in Fig. 5d. At the start of this scenario, Dovom *et al.* [17] has performed efficiently with an average difference of +/- 6.78% till 70 attacks; however, the performance reduces after that CMACIDS gets the advantage of obtaining steady performance line. All the behaviors of the four algorithms from all four scenarios signify the fact that the features extracted by CMACIDS is accurate and updates accordingly. Also with new learning behavior detection rates are almost steady in all variations of CMACIDS. Therefore, even though the variations and types of attacks are increased gradually in the scenarios CMACIDS performs efficiently as compared to others. The reason that the other algorithms have not performed well is due to lack of efficient feature extraction process and lack of efficient learning mechanism, which are inherent in CMACIDS algorithm.

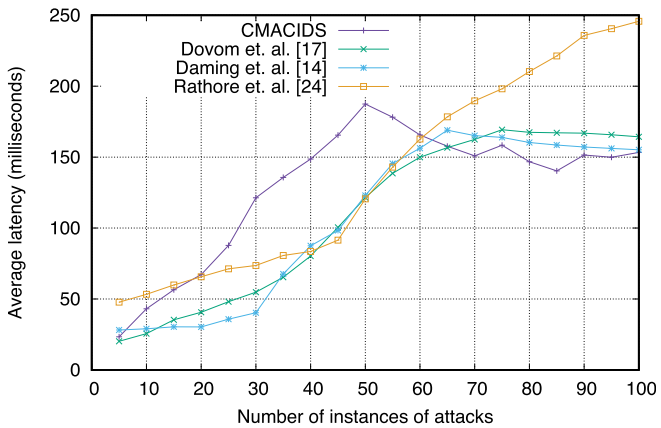


Fig. 6. Comparison of overall latency time including edge and cloud (Synergical latency comparison).

IoT deals with dynamic and real-time data and therefore, the latency or the detection time needs to be minimized for any kind of security implementation. The detection time and overall latency for all the approaches in comparison have been measured and shown in Table 6. We observe from the table that for some type of attacks, intrusion detection with CMACIDS is faster compared to other algorithms. However, IP sweep attack is detected faster in [24] and rootkit detection takes place faster in [14]. The results are shown in the table signify that the proposed approach is able to detect various attacks in less time as compared to other algorithms due to the timing advantages of CMAC process and kernel extractions. To emphasize and precisely monitor the algorithms in

TABLE 7
Complexity Comparison

	Computation complexity	Memory complexity
Dovom *et al.* [17]	$O(n^2+m)$	$O(n^2)$
Daming *et al.* [14]	$O(nlogn)$	$O(m^2)$
Rathore *et al.* [24]	$O(n^2)$	$O(mlogn)$
Proposed CMACIDS	$O(nm)$	$O(nm)$

n is the number of network features and is the number m of attacks features.

their detection capability, the latency time with the synergy of the multiple numbers of attacks at a time are compared. The results are shown in Fig. 6.

It shows that CMACIDS is better in providing low latency in the attack scenario and overall latency is also less as compared to the other algorithms. An interesting fact is observed from the results. At the initialization of the detection process, CMACIDS is consuming more time resulting in a high latency as compared to the other algorithms; however, with the increasing time and attack events the learning mechanism is adapted; the process of detection speeds up with reducing the detection time by 11.33% and overall latency by 23.67 % which is very significant in IoT applications. Being applied at the edge computing, time complexity and memory complexity need to be minimized. These two parameters are measured for all the algorithms and compared in Table 7 The complexity shows that the proposed approach exhibits very low complexity as it is linear due to the inclusion of CMAC with Spline adaptability, therefore it is suitable for IoT applications.

The stability of a system is important as it prevents the system from deviating from the standard mean error obtaining the nominal behavior of the scenarios. In our experiment, we measure the number of weight adjustments for the attack detection; less the number of weight changes signifies that the system is more stable. We compare the stability of CMACIDS with the other algorithms and summarize in Fig. 7. As can be seen from the Fig. 7, CMACIDS is more stable in terms of weight adjustments. In the presence of overall 100 various attacks, it achieves stable weight adjustments at 70 attacks and maintains it afterward; whereas the other algorithms are unstable as the weight adjustment is high and the graph follows an unsteady line over the entire range of attacks. Though Dovom *et al.* [17] is also having similar kind of weigh adjustment numbers after 40 attacks CMACIDS outperforms it completely. Finally, we show a comparison table in Table 8. In this table, we compare the major attributes of CMACIDS with the existing protocols. Finally, we compare the detection accuracy of all these attacks for all the approaches under consideration. We show this comparison in Table 9.

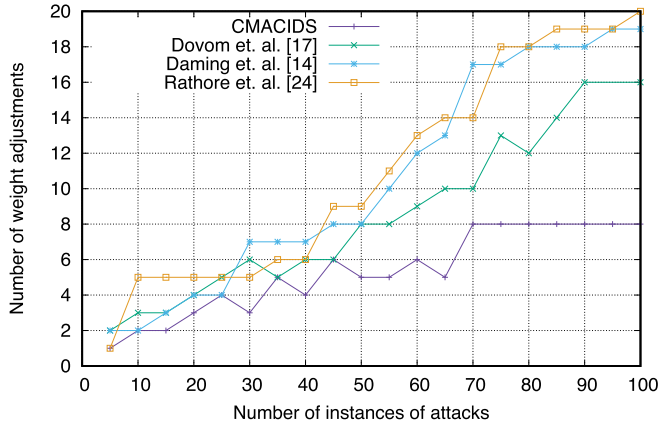


Fig. 7. Comparison of weight adjustments for stability.

5 Conclusion and Future Works

An automated and learning-based intrusion detection method, CMACIDS has been developed for IoT based communication networks. The method uses CMAC and spline fitting, respectively, for the learning the kernel map feature

TABLE 8
Feature Comparison

	Complexity	System stability	Memory usage	Pipelining	Learning rate
Dovom *et al.* [17]	High	Low	Non-linear	No	Tuning required
Daming *et al.* [14]	Average	Low	Non-linear	No	Tuning required
Rathore *et al.* [24]	High	Average	Non-linear	No	Tuning required
Proposed CMACIDS	Low	High	Linear	Yes	Tuning not required

TABLE 9
Comparison of Accuracy for Attack Types

	Dovom *et al.*, [17]	Daming *et al.* [14]	Rathore *et al.* [24]	Proposed CMACIDS
Sweep attacks	98.6%	98.8%	98.87%	100%
SPY attacks	93%	89.9%	95%	98.9%
IMAP attacks	90.33%	91.23%	92.55%	97%
Buffer overflow	99.37%	98%	98%	100%
Rootkit	93.37%	95.7%	98%	100%
Multihop	89%	92%	95%	95%

The values are calculated for all the instances and mentioned here in average only.

extracts and system adaptability. The mathematical model is less complex and follows a linear complexity derivation. The evaluation parameters and their comparative analysis show that CMACIDS increases the accuracy by 18.13% on average, reduces the latency by 23.67% and improves the stability by 11.8% in weight adjustments. To summarize, CMACIDS serves as an efficient solution for edge computing in the perception layers of the IoT. CMACIDS is generic; therefore, it is applicable in all scenarios of edge computing. The applicability of the model depends on the feature extraction from the network traffic. As an extension to our present work, various feature extraction methods can be applied to compare the extracted features. The ability of CMACIDS for detection of distributed attacks and advanced authentication attacks with the creation of authentication vector dictionary are to be followed as future direction of research.

REFERENCES

[1] J. H. Nord, A. Koohang, and J. Paliszkiewicz, "The Internet of Things: Review and theoretical framework," *Expert Syst. Appl.*, vol. 133, pp. 97–108, 2019.

[2] V. Tsiatsis, S. Karnouskos, J. Höller, D. Boyle, and C. Mulligan, "Chapter 1 - Why the Internet of Things?," in *Internet of Things*, 2nd ed., V. Tsiatsis, S. Karnouskos, J. Höller, D. Boyle, and C. Mulligan, Eds., Cambridge, MA, USA: Academic Press, 2019, pp. 3–7.

[3] E. Ekudden, "Exponential capability growth. Exponential potential," 2021. Accessed: Nov. 22, 2021. [Online]. Available: https://www.ericsson.com/en/blog/2021/10/exponential-capability-growth-exponential-potential

[4] M. Kraeling and M. C. Brogioli, "13 - Internet of Things," *Software Engineering for Embedded Systems*, 2nd ed., R. Oshana and M. Kraeling, Eds., Amsterdam, Netherlands: Newnes, 2019, pp. 465–499.

[5] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of Things security: A top-down survey," *Comput. Netw.*, vol. 141, pp. 199–221, 2018.

[6] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of Things applications: A systematic review," *Comput. Netw.*, vol. 148, pp. 241–261, 2019.

[7] P. P. Ray, "A survey on Internet of Things architectures," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 30, no. 3, pp. 291–319, 2018.

[8] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[9] D. Cotroneo, A. Paudice, and A. Pecchia, "Empirical analysis and validation of security alerts filtering techniques," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 5, pp. 856–870, Sep./Oct. 2019.

[10] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 4, pp. 198–211, Jul./Aug. 2013.

[11] M. Hasan, M. M. Islam, M. I. Islam Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, 2019, Art. no. 100059.

[12] A. P. Kelton *et al.*, "Internet of Things: A survey on machine learning-based intrusion detection approaches," *Comput. Netw.*, vol. 151, pp. 147–157, 2019.

[13] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "Intrusion detection in smart cities using Restricted Boltzmann Machines," *J. Netw. Comput. Appl.*, vol. 135, pp. 76–83, 2019.

[14] D. Li, L. Deng, M. Lee, and H. Wang, "IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning," *Int. J. Inf. Manage.*, vol. 49, pp. 533–545, 2019.

[15] S. Deshmukh-Bhosale and S. S. Sonavane, "Real-time intrusion detection system for wormhole attack in the RPL based Internet of Things," *Procedia Manuf.*, vol. 32, pp. 840–847, 2019.

[16] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Comput. Netw.*, vol. 159, pp. 96–109, 2019.

[17] E. M. Dovom, A. Azmoodeh, A. Dehghantanha, D. E. Newton, R. M. Parizi, and H. Karimipour, "Fuzzy pattern tree for edge malware detection and categorization in IoT," *J. Syst. Archit.*, vol. 97, pp. 1–7, 2019.

[18] Y. Guo, H. Zhang, L. Zhang, L. Fang, and F. Li, "A game theoretic approach to cooperative intrusion detection," *J. Comput. Sci.*, vol. 30, pp. 118–127, 2019.

[19] S. Halder, A. Ghosal, and M. Conti, "Efficient physical intrusion detection in Internet of Things: A Node deployment approach," *Comput. Netw.*, vol. 154, pp. 28–46, 2019.

[20] W. Li, S. Tug, W. Meng, and Y. Wang, "Designing collaborative blockchained signature-based intrusion detection in IoT environments," *Future Gener. Comput. Syst.*, vol. 96, pp. 481–489, 2019.

[21] Z. Pan, S. Hariri, and J. Pacheco, "Context aware intrusion detection for building automation systems," *Comput. Secur.*, vol. 85, pp. 181–201, 2019.

[22] R. Yahalom, A. Steren, Y. Nameri, M. Roytman, A. Porgador, and Y. Elovici, "Improving the effectiveness of intrusion detection systems for hierarchical data," *Knowl.-Based Syst.*, vol. 168, pp. 59–69, 2019.

[23] A. Shenfield, D. Day, and A. Ayesh, "Intelligent intrusion detection systems using artificial neural networks," *ICT Exp.*, vol. 4, no. 2, pp. 95–99, 2018.

[24] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for IoT," *Appl. Soft Comput.*, vol. 72, pp. 79–89, 2018.

[25] H. Moudni, M. Er-rouidi, H. Mouncif, and B. El Hadadi, "Black hole attack detection using fuzzy based intrusion detection systems in MANET," *Procedia Comput. Sci.*, vol. 151, pp. 1176–1181, 2019.

[26] D. Jianjian, T. Yang, and Y. Feiyue, "A novel intrusion detection system based on IABRBFSVM for wireless sensor networks," *Procedia Comput. Sci.*, vol. 131, pp. 1113–1121, 2018.

[27] X. Jinhui, T. Yang, Y. Feiyue, P. Leina, X. Juan, and H. Yao, "Intrusion detection system for hybrid DoS attacks using energy trust in wireless sensor networks," *Procedia Comput. Sci.*, vol. 131, pp. 1188–1195, 2018.

[28] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Comput. Secur.*, vol. 77, pp. 304–314, 2018.

[29] D. Santoro, G. Escudero-Andreu, K. G. Kyriakopoulos, F. J. Aparicio-Navarro, D. J. Parish, and M. Vadursi, "A hybrid intrusion detection system for virtual jamming attacks on wireless networks," *Measurement*, vol. 109, pp. 79–87, 2017.

[30] L. Han, M. Zhou, W. Jia, Z. Dalil, and X. Xu, "Intrusion detection model of wireless sensor networks based on game theory and an autoregressive model," *Inf. Sci.*, vol. 476, pp. 491–504, 2019.

[31] S. Shen, Y. Li, H. Xu, and Q. Cao, "Signaling game based strategy of intrusion detection in wireless sensor networks," *Comput. Math. Appl.*, vol. 62, no. 6, pp. 2404–2416, 2011.

[32] A. Hassanzadeh, Z. Xu, R. Stoleru, and G. Gu, "Michalis polychronakis, PRIDE: A practical intrusion detection system for resource constrained wireless mesh networks," *Comput. Secur.*, vol. 62, pp. 114–132, 2016.

[33] A. Hassanzadeh, A. Altaweel, and R. Stoleru, "Traffic-and-resource-aware intrusion detection in wireless mesh networks," *Ad Hoc Netw.*, vol. 21, pp. 18–41, 2014.

[34] A. Hassanzadeh and R. Stoleru, "On the optimality of cooperative intrusion detection for resource constrained wireless networks," *Comput. Secur.*, vol. 34, pp. 16–35, 2013.

[35] A new approach to manipulator control: The cerebellar model articulation controller. Accessed: Jan. 20, 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/sp958-lide/237–240.pdf

[36] Y. Jiang and Y. Li, "The research of the approximate algorithm based on cubic b-spline curves," in *Proc. 5th Int. Conf. Inf. Comput. Sci.*, 2012, pp. 23–26.

[37] H. A. Khattak, M. A. Shah, S. Khan, I. Ali, and M. Imran, "Perception layer security in Internet of Things," *Future Gener. Comput. Syst.*, vol. 100, pp. 144–164, 2019.

[38] O. Valdez-de-Leon, "Key elements and enablers for developing an IoT ecosystem," *IEEE IoT Blog*, May 17, 2017. [Online]. Available: https://iot.ieee.org/newsletter/may-2017/key-elements-and-enablers-for-developing-an-iot-ecosystem.html
[39] I. Lee, "The Internet of Things for enterprises: An ecosystem, architecture, and IoT service business model," *Internet Things*, vol. 7, 2019, Art. no. 100078.
[40] A. R. Teixeira, A. M. Tomé, and E. W. Lang, "Unsupervised feature extraction via kernel subspace techniques," *Neurocomputing*, vol. 74, no. 5, pp. 820–830, 2011.
[41] A. Hinrichs, J. Prochno, and M. Ullrich, "The curse of dimensionality for numerical integration on general domains," *J. Complexity*, vol. 50, pp. 25–42, 2019.
[42] C.-M. Vong, C. Chen, and P.-K. Wong, "Empirical kernel map-based multilayer extreme learning machines for representation learning," *Neurocomputing*, vol. 310, pp. 265–276, 2018.
[43] R. MacAusland, "The moore-penrose inverse and least squares," 2014. [Online]. Available: http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-macausland-pseudo-inverse.pdf
[44] W. Wan, "Implementing online natural gradient learning: Problems and solutions," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 317–329, Mar. 2006.
[45] T. Ahmed, M. Coates, and A. Lakhina, "Multivariate online anomaly detection using kernel recursive least squares," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2007, pp. 625–633.
[46] E. T. Y. Lee, "Comments on some B-spline algorithms," *Computing*, vol. 36, pp. 229–238, 1986.