GMR
The Original

A DISCRETE-OPTIMIZATION WITH HIERACHICAL CLASSIFIER FOR ATTACK PREDICTION

Hari Kishore Chejarla[1], Kiran K.V.D[2]

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram522302, Andhra Pradesh, India; Email: harichejarla@gmail.com,
[2]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram 522302, Andhra Pradesh, India; Email: kiran_cse@kluniversity.in

ABSTRACT
Machine learning techniques are designed to act as a buffer against attack like jamming and cloning that are initiated over wireless networks. A pre-trained classifier is provided to the transmitter so that it may assess the channel's condition depending on the type of sensing it is doing and decide what needs to be transmitted next. All acknowledgements made among the nodes and the current channel state are gathered by the learning approach to construct a learning model that properly predicts the consecutive transmission limitation caused by network jamming. Here, the goal of an inventive anti-clone detection approach is to diminish the quantity of jamming and clones found throughout the network model relative to stochastic jamming methods. The transmitter analyzes the power restrictions over the sensor networks using the Discrete Particle Swarm Optimization (DPSO). In this case, by analyzing the incoming samples, a Hierarchical Prototype-based classifier is modelled to reduce the computing time required to gather the training dataset. By using this defence mechanism, the broadcaster hopes to forecast the false prediction rate (FPR) and create a more accurate model that will produce a classifier that is dependable. To increase throughput and lower prediction error, the transmitter systematically detects floating attack over the network model and implements the defence mechanism to trick the injected clone.

KEYWORDS- Wireless Sensor Networks, clone, jamming, adversarial network model, fisher score, classifier

1. INTRODUCTION

Information systems are an important tool that businesses and organizations may utilize in the present era of modernization to effectively acquire, and organize the way you handle, organize, store, and share information. Since businesses have started using information system technologies, there has been a chance of security breaches. Organizations have implemented several information security procedures to defend sensitive data and the Information Technology (IT) infrastructure against online threats [1]. The intrusion detection system is one of the most important parts of cybersecurity (IDS). An IDS can detect unusual traffic and notify the network administrator of the information it receives. According to [2], Four functional modules are often the foundation of an intrusion detection system's (IDS) architecture: (1) Event; (2) Database; (3) Analysis; and (4) Response. To put it briefly, the Eventis made up of sensor components that are employed to keep an eye on the system and collect event data for further examination. For processing, it is necessary to store the collected data. The information originating from the event block is stored in the data block components for this reason [3]. The processing module is a block where events are analyzed to detect harmful behavior. Preventing the danger is the most crucial duty after identifying the hostile conduct. Accordingly, in the case of an intrusion of any sort, to counteract the damaging occurrence, According to [4], the response boxes, often known as the response block, take the initiative to perform the appropriate reaction. This is the IDS architecture [5]. Depending on the information source, an IDS can be categorized as a host-based IDS (HIDS) or a network-based IDS (NIDS) [6].System calls and process IDs are connected to operating system information in HIDS, NIDS, on the other hand, analyzes network events, for example, traffic volume, service ports, protocols, and IP addresses [7]. IDS may be separated into two groups: IDS based on signatures (misuse-based) and IDS based on anomalies, depending on the analysis done.A database of recognized attack signatures is kept up to date in signature-based intrusion detection systems (SIDS), and the intrusion is detected by comparing the examined data with the database [8]. It works best at identifying known attack; nevertheless, it cannot identify novel or undiscovered attacks. Compared to signature-based IDS, The rate of false positives is considerably reduced [9]. Conversely, Anomaly-based IDS (AIDS) establishes a threshold value while attempting to comprehend the system's typical behaviour. An anomaly warning is triggered when the provided observation at any particular moment deviates from typical behaviour by more than the predetermined threshold value [10]. An excellent way to identify attack that hasn't been observed before is via an anomaly-based intrusion detection system (IDS). However, compared to SIDS, AIDS has a higher false-positive rate of detecting intrusions [11] – [12].

Using the Internet of Things (IoT) and cloud services is widespread that there is a daily spike in network traffic. IDS finds it difficult to discern between typical and unusual network traffic patterns, particularly in the identification of zero-day attacks (attacks that have not been previously discovered) [13]. ML approaches have emerged as a practical and efficient means of classifying and identifying various forms of network attacks in order to address this problem. Machine learning technology allows the IDS to learn and improve system performance by analyzing past data. Furthermore, because machine learning systems may learn on their own, they don't need to be explicitly coded. Given that machine learning models' detection rate and accuracy score are insufficient to properly classify the breach, the intrusion detection approach remains at the centre of many types of research activities. Moreover, the author in [14] found that many strategies are insufficiently effective while using the complete dataset, for a sizable amount of data. Additionally, a lot of intrusion detection research has been done on the NSLKDD 99 or KDD 99 dataset, which is no longer thought to be up to date for identifying contemporary cyber-attacks [15].

This study used supervised machine learning models to carry out a binary classification job.
When a discrete value is allocated to be predicted as either a "normal" or "attack" instance by a supervised machine learning model, binary classification occurs [16]. This study's dataset is quite huge and has a high feature space dimension. Choosing a feature selection approach that eliminates unnecessary characteristics from a big dataset is crucial, because it allows the training and testing stages to employ just the pertinent collection of characteristics [17].Feature selection is the process of choosing a subset of relevant qualities in order to reduce the overall number of input variables needed to build a predictive model. Most of the time, improving the performance of predictive models is the goal. Furthermore, superfluous features detract from the model's performance. However, the IDS increases the processing cost in addition to lowering the detection accuracy score since it handles a large number of data instances with redundant or irrelevant attributes [18]. By selecting the pertinent features that include the necessary data for the classification process, feature selection provides answers to several often made errors in IDS.

A weighted Random Forest (GIWRF) model based on Gini Impurity was created in our suggested method to choose the pertinent and significant features according to the significance score. To account for the unbalanced class distribution, the Random Forest method [19] was adjusted for weight, and the trees were divided using the Gini impurity criteria to determine the feature significance scores. To develop an ML-based IDS framework, two datasets were used to construct the Decision Tree (DT), Gradient Boosting Tree (GBT), Multilayer Perceptron (MLP), Adaptive Boosting (AdaBoost), Long-Short Term Memory (LSTM), and Gated Recurrent Unit (GRU). Before the models were trained and tested, data pre-treatment or data engineering was necessary. Three phases were involved in the data preparation phase, and these will be covered later in this work. Nonetheless, the UNSW-NB 15 dataset was used to undertake performance analysis regarding the challenge of binary categorization. Network IoT dataset [20] are relatively fresh datasets that contain contemporary attack patterns. The major research contributions are listed below:

1) The experimentation is carried out using online-available dataset where DPSO and Hierarchical prototype-based classifier is proposed;
2) The classifier performs self-learning and self-evolution in a hierarchical manner based on the streaming attack dataset;
3) The model poses an ability to deal with complex issues and mine essential information in an efficient manner for attack prediction.
4) Some metrics like precision, accuracy, recall and F-measure are evaluated and compared with other approaches.

The work is presented as: The literature is elaborated in section 2 with methodological analysis in section 3. The outcomes are represented in section 4 and summary is provided in section 5.

2. RELATED WORKS

IDS may be divided into two groups: anomaly detection and abuse detection. Signatures, which are established patterns of known attack, are used in misuse detection. According to [21], it may be further separated into state-full and stateless IDS. The state-full methods use both current and historical signatures, whereas state-less approaches just rely on the existing signature [22]. For known attack, this method offers excellent precision and low rates of false alarms; nevertheless, it is impractical for identifying unknown attacks [23]. Updating the database regularly is one of the recognized remedies to this issue, but it is an expensive and time-consuming procedure. Therefore, it is not seen as possible. By contrast, user behaviour profiling is the focus of anomaly detection [24]. This method defines a certain model of typical user behaviour, and any departure from the model is referred to be abnormal. Static and dynamic approaches are other categories into which anomaly detection techniques may be divided [25]. The fixed part of the system can only be used with the static anomaly detection approach. Using network usage history, the dynamic anomaly detection approach creates "profiles"—patterns—

from the data. Although this technique lacks high accuracy and may result in high FAR, it can detect fresh attack[26]. This tactic also has the disadvantage of making an attacker feel as though he is being profiled. The author may gradually revert the system's anomalous behaviour to normal. To further enhance intrusion detection, researchers have also recently proposed hybrid techniques [27].

Based on their design, IDSs may be categorized into three groups: Host, network, and hybrid intrusion detection systems (IDSs; a mix of host and network). Every computer system that is a component of a host-based intrusion detection system has an agent or sensor installed on it [28]. It analyzes system calls, audit trails, application logs, and other host activity to find intrusions. It is the developer's responsibility to alter the operating system kernel code if more event data or logs are required. Some clients may find this strategy undesirable as it raises costs [29]. Furthermore, it might be laborious to install the agent across all computer systems.
IDS is set up on the server in the network-based system. By tracking network traffic across several hosts, the sensors are used to detect intrusions [30]. They are very portable, simple to use, and operating system-agnostic. It does, however, exhibit limits when high-speed data or network traffic surges are present. IDS must be installed on both the server and every client in a hybrid system. It is regarded as the most sensible and successful method for intrusion detection as it integrates host and network techniques [31].

Businesses employ signature-based intrusion detection systems (IDS) to defend themselves against known attack, the signatures of which are stored in a database. This IDS search assessed the pattern by comparing it to a list of known patterns and harmful bytes. IDS based on signatures can convey the reason for an intrusion alarm [32]. When it comes to novel attack, signature-based intrusion detection systems are less effective because they cannot identify patterns in them or keep up with database updates. This problem can be resolved with routine database changes of patterns [33]. However, signature-based detection is not effective when the user mounts attack using sophisticated technologies such as payload encoders, encrypted data channels, and no-operation (NOP) generators [34]. Create a fresh signature for each alteration results in a considerable reduction in its efficiency. Additionally, the system engine's performance degrades as the number of signatures rises. The reasons for working in the anomaly detection IDS sector are the inability to identify new attacks and to routinely refresh the database for new patterns [35]. Table 1 depicts the performance comparison of various prevailing approaches.

Table 1 Comparison of prevailing approaches

References	Dataset	Method	Merits	Demerits
[16]	KDD-CUP 1999	SVM, PCA	The RBF kernel outperforms polynomial kernel-based SVM in terms of detection time and DR.	--
[18]	NSL-KDD	Association between Self-Taught Learning (STL) and (Sparse autoencoder, Soft-max)	85.44% is the 2-class precision. 95.95% of the recall 90.4% is the F-Measure. 88.39% accuracy	It is necessary to build real-time NIDS with more effective and dynamic feature learning.
[20]	KDD 99	SAE is used as a Feature Extraction and Selection Method	F-Measure for Class 5 = 75.76% 79.10% accuracy Class-wide accuracy is 98%.	Changes may be made to parameters like the size of the batch, learning rate, and number of iterations. Kim and Aminanto (2017) The feature extraction and selection process uses AWID SAE. Technique Rate of Detection = 92.18% FAR is 4.40%. 94.81% accuracy Accuracy = 86.15% F1 is 89.06%. SAE will be utilized in the future for unknown assault identification as an outlier detection method.
[24]	NSL-KDD	Information gain with k-means clustering	Accuracy = 99.4% Precision = 99.8%	--

[28]	KDD-CUP 1999	ANN and RNN-IDS comparison J48, placePlaceNameRandomPlaceTypeForest, SVM	Rate of Detection = 92.18% FAR is 4.40%. 94.81% accuracy Accuracy = 86.15% F1 is 89.06%.	Sampling techniques can be employed to improve the classification system's performance .
[30]	NSL-KDD	ANN SVM and CFS-ANN-based classifier	Accuracy rates for two-class classification are 93.2% and for multi-class classification are 54.13%.	The next objective is to use GPU acceleration to shorten the training time.
[35]	UNB-CIC	k-NN Softmax Regression in PCA Model	DR 100%, FPR 1.2%, and accuracy of 99.8% are achieved in detecting non-Tor traffic.	Following the classification of eight separate traffic categories in the UNB-CIC Tor Network Traffic dataset, a performance analysis will be conducted later.
[19]	KDD Cup 99	Deep Feature extraction and Selection, or D-FES, is utilized as a clustering method	Softmax regression demonstrated improved time efficiency.	Pre-processing methods can be used to manually optimize the number of features in a reduced feature set.

3. METHODOLOGY

This study's primary goals were to compare the functionality of the machine learning models that were highlighted in the section ML techniquesand to use an acceptable feature selection methodology for unbalanced data after they were trained on datasets, compiled in the benchmark dataset section. Fig 1 shows the schematic for the suggested methodology.

3.1. DPSO optimization

Particle Swarm Optimization (PSO) is a population-based optimization method that was developed recently. It is based on the social behavior of bird flocks and fish schools. Kennedy and Eberhart were the first to use PSO to optimize a variety of continuous nonlinear functions. The optimal operational pathfinding for automated drilling operations, travelling salesman problems, supplier selection and ordering problems, neural network training, mass-spring system, power and voltage control, task assignment and other issues have all been successfully resolved by PSO as a result. The literature does not contain any reports on the application of the DPSO method to scheduling issue resolution.The PSO iteratively explores the problem space for the best solution after being given a population of randomly generated starting solutions. Particles, or possible solutions, are identified by their best particles and fly across the issue space. PSO uses a collection of randomly generated sequences known as particles to begin the search process. P_{best}andG_{best} sequences are then determined after that.

Intrusion detection framework

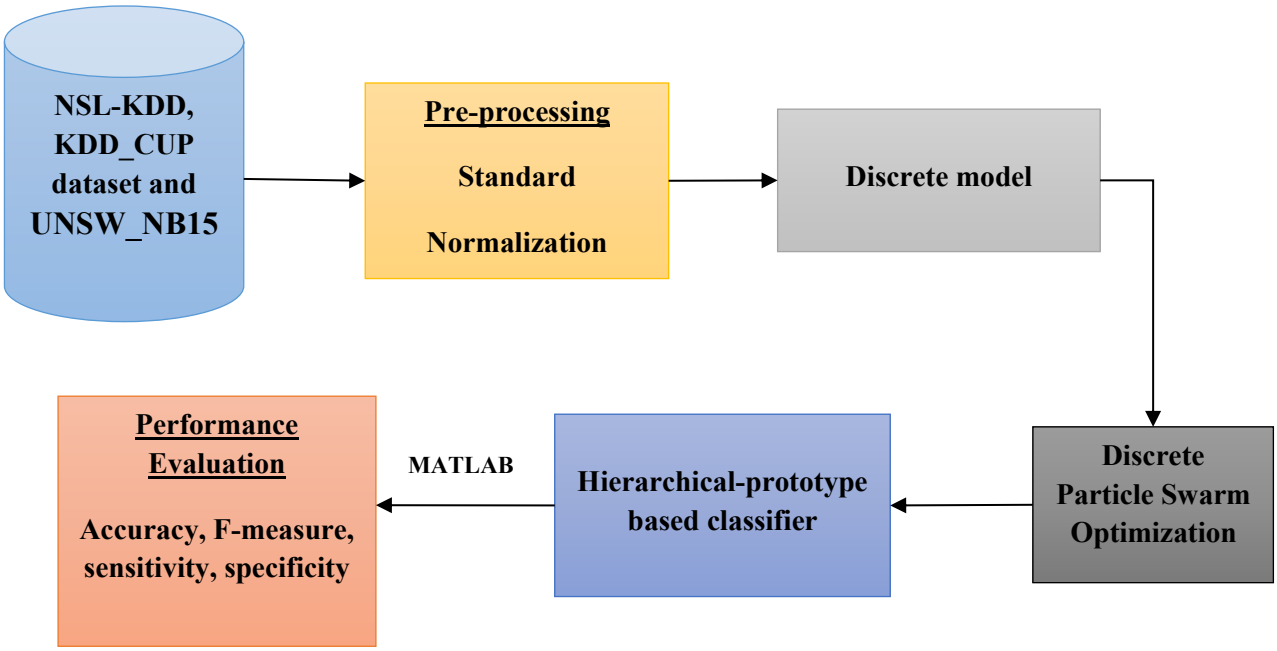

Fig 1 Block of the proposedmodel

Using Eq. (1), the iterative search procedure now begins with calculating the velocity of each particle, v_k.

$$v_k^{t+1} = \ c_1 * v_k^t + c_2(P_{best} - P_{current}) + c_3\ (G_{best} - P_{current}) \quad (1)$$

The present velocity of a particle,v_k^t, as well as the $P_k(best)$ and G_{best} sequences, define its new location at time $(t+1)$. The coefficients of learning are $c_1, c_2, and\ c_3$. Eq. (2) is utilized to update the particle position or sequence, following each repetition. Particle velocity plus current position equals new position.

$$P_k^{t+1} = P_k^t + v_k^{t+1} \quad (2)$$

Eq. (3) is used to calculate a particle's velocity index at time t.

$$v_k = \left((i_q, j_q)\right);\ q = 1, L \quad (3)$$

where, job positions are represented by the variables $i, j, and\ L$ which stand for the list's length or the maximum number of transpositions that can occur between 1 and n. The exchange of work positions (i_1, j_1) and so on is denoted by v_k in this instance. Eq. () is used to calculate the particle velocity,v_k^t, once the velocity index,v_k is generated. The following describes the many procedures used to update particle locations and calculate particle velocity. Assumethat, two locations, x_1 and x_2 correspond to two different sequences. Use the subtraction $(P_{best} - P_{current})$ operator to get the result. Velocity v is the difference between $x_1$and x_2. For instance, removing two positions or $(P_{best} - P_{current})$ in Eq. (4) yields a velocitywhich is a group of substitutions.Let x and v be the addition operators for position plus velocity to express position and velocity. Applying v first transposition to $p\ (x_1 = x + v)$, followed by $v's$ second transposition to the result yields the new position x_1. Operator for addition (velocity + velocity): Let $v_1\ and\ v_2$ be the two velocities. To compute $v_1 + v_2$, we evaluate a list of transpositions, and the first and second 'ones' of $v_1\ and\ v_2$, respectively, are those locations.In the multiplication operator, let v represent velocity and c the learning coefficient (Coefficient × velocity). As a result, the velocity $c \times v$ is created. This section provides a numerical representation of the previously stated steps utilized in the suggested DPSO method.

Implementation: For the t^{th} iteration, let us assume the following information. the current sequence $(P_k^t = \{2,3,1,4\})$; the best sequence $(P_k^t(best) = \{2,3,1,4\}\ (P_{best}\ Sequence)$; and the greatest sequence $(G^t = \{3,4,2,1\})$ The following describes the computation process used to determine particle velocity and movement:

Generating Velocity Index: The particle $k's$ velocity index, v_k is calculated as follows: Set the length of the velocity list L_k to two and use the randomly generated list $\{(2,4),(4,1)\}$. For other particles, the velocity index is calculated in the same way.

Calculation of Particle Velocity: Eq. (4) is used to determine the velocity needed to move each particle in the population from one location to another. The quantity of velocity components that must be used throughout the location is determined by multiplying the coefficient by the velocity operator. In the event where the coefficient value is 0.5, for instance, half of the velocity components are applied over the position and chosen at random from the velocity list.

$$v_k^{t+1} = [(2,4),(1,2),(2,4)] \tag{4}$$

Particle Movement: The particles are moved from their current location to the new one using Eq. (5):

$$P_k^{t+1} = P_k^t + v_k^{t+1} \tag{5}$$
$$P_k^{t+1} = \{2,3,4,1| \tag{6}$$

In a similar manner, P_k^{t+1} is calculated for each particle and $G\ (G_{best})\ and\ P_k^t(best)\ (P_{best})$ are updated.

3.2. Hierarchical-Prototype-Based-Classifier

This section provides a detailed presentation of the proposed HP classifier's broad learning, decision-making and architecture processes. The computational complexity of the recommended approach is also looked. Given that,$T \in R_N, let\ x_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,N}]$. Let us first consider a specific data stream in the $N-$dimensional real data space, R^N, denoted by $\{x\} = \{x_1, x_2, \ldots, x_k, \ldots, x_K, \ldots\}$. The subscript k indicates the time occurrence at which x_k is seen. The observed data samples from the data stream $\{x\}$ at the K^{th} time instance can be divided into C subsets according to their class labels since there are C distinct classes in the stream: $K = \sum_{i=1}^{c} K^i$ is indicated by the formula $\{x_K^i = \{x_1^i, x_2^i, \ldots, x_{K_i}^i\}$, where $i = 1, 2, \ldots$, and the superscript i indicates the i^{th} class. Unless otherwise indicated, any mathematical derivations made in the next portions of this article are done so by default at the K^{th} time instance.

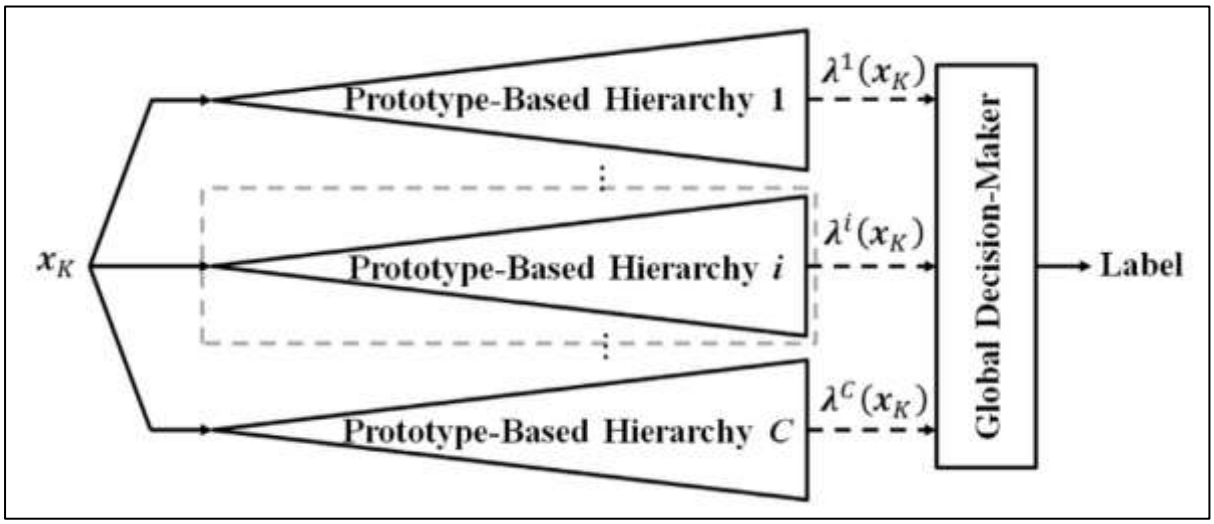

Fig 2 Classifier model

The HP classifier's general design is displayed in Fig 2. The figure shows the C distinct prototype-based pyramidal hierarchies that comprise the HP classifier, one for each of the C classes that are present in the data stream. During learning, "one-pass" self-organizing data samples from the appropriate class are used to train each hierarchy simultaneously.$j = 1, 2, \ldots, M_l^i, and\ p_{l,j}^i$signifyingthe zoomed-in structure of the i^{th} hierarchy is shown in Fig 3 which shows the j^{th} prototype at the l^{th} tier of the i^{th} hierarchy which is generated from data samples of the i^{th} class with $l = 1, 2, \ldots, L$ as the layer number. Here,$M_1^i \leq M_2^i \leq \ldots \leq M_{L-1}^i \leq M_L^i;\ 1 \leq m_1 \leq m_2 \leq M_2^i;\ 1 \leq m_3 \leq M_{L-1}^i;\ 1 \leq m_4 \leq m_5 \leq m_6 \leq m_7 \leq M_{L-1}^i.M_L^i$represents the quantity of prototypes at the l^{th} layer. Without compromising generality, the HP classifier prototype-based hierarchies used in this work all share the

same layer (L). In the C pyramidal hierarchies, prototypes are created. "one pass," top-down, directly from the data. Higher up the hierarchical ladder, these prototypes are more abstract and representational and they provide more broad information. Lower-level prototypes, on the other hand, are more intricate and resemble the first observed data samples inside the data space. A prototype is a local maximum of its immediate superior's data's multimodal distribution. Every prototype at a layer is connected to a single immediate superior prototype at the layer above, with the exception of apex prototypes at the top layer. All prototypes, with the exception of leaf prototypes at the lowest layer, are continually linked to one or more immediate subordinate prototypes at the layer below.$P_{1,1}^{i}$, for instance, is the immediate superior of $p_{2,1}^{i}, \ldots, p_{2,m_1}^{i}$ as shown in Fig 3, whereas $P_{L-1,1}^{i}$ are the immediate subordinates of $p_{L-1,1}^{i}$.

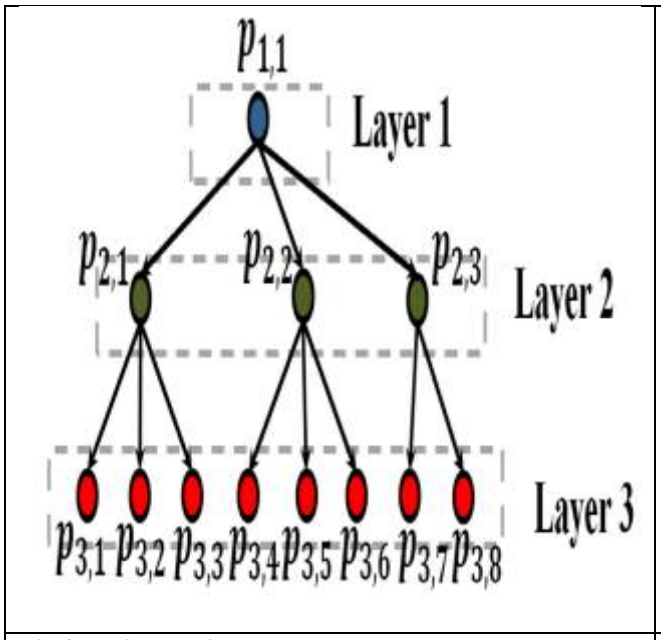

Fig 3a. Hierarchical layers

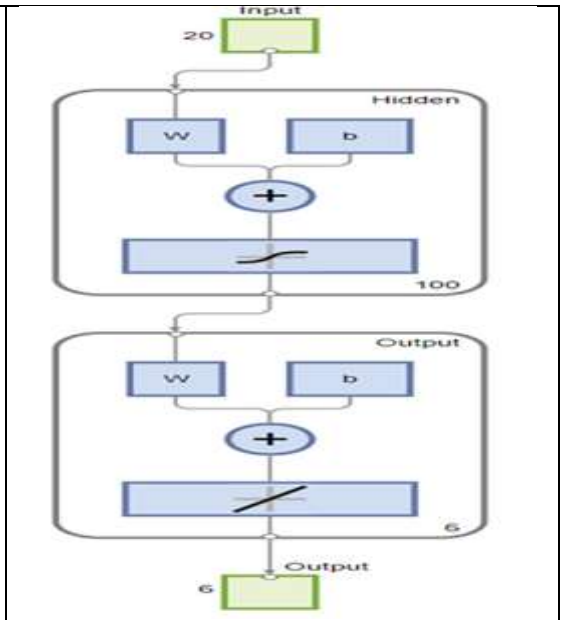

Fig 3b. Intermediate node functionality

3.3. Learning process

This sub-section provides a detailed description of the algorithmic procedure for independently determining a hierarchical prototype-based system structure. Just the i^{th} prototype-based hierarchy's learning process ($i = 1, 2, \ldots, C$)is provided because the suggested approach uses training samples from each class to independently identify prototypes and establish a top-down, hierarchical pyramid on their own. The base of any other hierarchy within the system can be formed by the same ideas. By default, the Euclidean norm of each observed data sample of the i^{th} class, $x_k^i(k = 1, 2, \ldots, K^i, \ldots)$, is first applied.

$$x_k^i \rightarrow \frac{x_k^i}{||x_k^i||} \tag{7}$$

where,$\left|\left|x_k^i\right|\right| = \sqrt{\sum_{j=1}^{N}\left(x_{k,j}^i\right)^2}$. This kind of normalization increases the HP classifier's performance by converting the cosine dissimilarity-based distance measure from the Euclidean distance between data samples to handle high-dimensional complex scenarios.

Initialization: The initial seen data sample of the i^{th} class, $x_{K^i}^i(K^i = 1)$, is used to construct the hierarchy and is also regarded as the prototype at each layer ($l = 1, 2, \ldots, L$):

$$M_l^i \rightarrow 1 \ and \ S_{l,M_l^i}^i \rightarrow 1 \tag{8}$$

where, $S_{l,M_l^i}^i$ is the number of data samples related to $P_{l,M_l^i}^i$. The ties (subordinate relationships) between these prototypes of later levels are then established, forming a hierarchy.

$$L_o^i \rightarrow \{p_{1,M_1^i}^i\} \tag{9}$$

To begin with, the set of apex prototypes is described as follows:

$$L^i_{l,M^i_l} \rightarrow \{P^i_{l+1}, M^i_{i+1}\} \tag{10}$$

Furthermore, the prototype gathers its immediate subordinates at the l^{th} layer, P^i_{L,M^i_L}. Eq. () is first used to construct the i^{th} hierarchy as a chain, with p^i_{1,M^i_1} serving as the starting node and P^i_{l+1}, M^i_{i+1} serving as the terminating node.

System evolution: With streaming data, the HP classifier may automatically update its meta- parameters and constantly self-evolve its system structure after setup. The system update procedure begins at the top layer ($l = 1$) and works its way down to the new data sample, $x^i_{K^i}$ ($K^i \rightarrow K^i + 1$), upon observation. First, using the following equation, the closest prototype at the l^{th} layer to $x^i_{K^i}$is found:

$$n^*_l = \begin{cases} argmin_{p \in L^i_0}(||x^i_{K^i} - p) & if\ l = 1 \\ argmin_{p \in L^i_{1-L,n^*_{l-1}}}(||x^i_{K^i} - p) & if\ l = 2,3,..,L \end{cases} \tag{11}$$

It is commonly recognized that when using a prototype-based strategy to learn about large-scale, complicated problems, discovering a high number of prototypes is frequently unavoidable. Eq. (11) considerably enhances the computing effectiveness of the nearest prototype search procedure by narrowing the search scope from the entire data space to a particular group of comparable prototypes. This can reduce the waste of computing resources because Most of the prototypes that may be discovered in the data space are spread out from $x^i_{K^i}$. When compared to other ways that also use the "nearest prototype" premise, the proposed HP methodology can find the closest prototype with very high efficiency thanks to this searching mechanism.Following the identification of the closest prototype $p^i_{l,n_{l^*}}$, the following criterion is examined to see if $x^i_{K^i}$ can emerge as new iteration of the lth layer prototype:

$$n^*_l = \begin{cases} argmin_{p \in L^i_0}(||x^i_{K^i} - p) & if\ l = 1 \\ argmin_{p \in L^i_{1-L,n^*_{l-1}}}(||x^i_{K^i} - p) & if\ l = 2,3,..,L \end{cases} \tag{12}$$

At the l^{th} tier of the hierarchy, the radius of the effect zone around each prototype is where, demonstrating an apparent and fascinating degree of closeness at the lth level of granularity. A larger difference than r_l between $x^i_{K^i}$ and $p \in L^i_{1-L,}$ indicates that $x^i_{K^i}$ provides a new data pattern that cannot be explained by any prototypes at the lth layer. $x^i_{K^i}$is therefore included as a new prototype in the system. In this work, the following equation is used by default to determine the value of r_l ($l = 1, 2, \ldots, L$).

$$r_1 = 2\ (1 - \cos\left(\frac{\theta_0}{2^{l-1}}\right)) \tag{13}$$

where,$\pi/2 = \theta_0$. The HP classifier may divide the data space at different granularities and learn from the data thanks to the radii of the subsequent layers' diminishing values. The broad picture of the issues may be seen by the higher levels of the pyramidal hierarchies, while more specific details can be seen by the lower tiers.The radii of influence areas, $r_l(l = 1, 2, \ldots, L)$, must be emphasized as being adjustable. Their values only need to adhere to the straightforward principle that $r_1 > r_2 > \ldots > r_l > \ldots > r_L$, indicating that prototypes at the top layer have a wider sphere of effect than those in the lowest stratum. Additionally, the radii, r_l ($l = 1, 2, \ldots, L$), may be estimated without any prior knowledge and are not specific to any particular user or situation. Conversely, users have the ability to determine their values by choice. Various approaches can be investigated to determine the values of$r_l(where\ l = 1, 2, \ldots, L)$ based on the particular requirements of the issues. Additionally, some existing work presents a different approach based on mutual distances that may be used to immediately generate $r_l(l = 1, 2, \ldots, L)$ from data sets that have been seen experimentally. The complexity of the issues and the available processing resources may also be used to obtain each hierarchy's tier number. However, this paper's primary goal is to provide the idea and broad concepts; as a result, only the generic settings are used.

4. NUMERICAL RESULTS AND DISCUSSION

This section provides a wider analysis of the experimentation carried out by the anticipated model.

4.1. Evaluation metrics

Increasing the percentage of accurate predictions in the binary classification test set was the aim of this investigation.A number of metrics for machine learning performance, including accuracy, recall, precision, false positive rate, and recall, are used to assess the effectiveness of ML-based intrusion detection systems. Nonetheless, the most often used statistic to assess a model's effectiveness in binary classification issues is its Accuracy (AC) score. It is defined by the equation below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

where, TN denotes the number of True Negative and TP stands for True Positive. The quantity of successfully identified assault cases is known as the True Positive. True Negative is the total number of correctly categorized normal cases. False Positive, or FP, on the other hand, is used to describe the quantity of error-prone.Since incursion more dangerous than FP, less FN is usually anticipated while detecting it. However, four more metrics are taken into account in addition to the accuracy measure:

FPR (False Positive Rate), FPR, Precision, Recall, and F1 score, especially for unbalanced class data where the objective is to accurately detect the incursion (positive occurrences).False Positive Rate (FPR): The ratio of incorrectly categorized negatively classed events that are incorrectly classified as positive (FP) relative to the total number of occurrences that are truly negative is known as the false positive rate (FPR). Here is an illustration of the term in action:

$$FPR = \frac{FP}{TP + FP} \tag{15}$$

Precision: A model's precision is a performance metric that shows how frequently a model successfully predicts an event when it makes an accurate forecast. It may be stated mathematically as the following equation:

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

Recall/Detection Rate (DR): It is a measure of the machine learning model's ability to identify True Positive situations. Recall also assesses how well the model finds pertinent information. It is also referred to as the True Positive Rate or Sensitivity for this reason. The following is an example of the mathematical expression:

$$DR = \frac{TP}{TP + FN} \tag{17}$$

F1-score: The ML model's total accuracy is determined by balancing recall and precision, This takes FN and FP into account. The following equation represents the F1 score:

$$F1 = 2 * \frac{Precision * DR}{Precision + DR} \tag{18}$$

4.2. Discussion

This section compares several machine learning and deep learning algorithms applied to benchmark datasets for the IDS. Recall, accuracy, and precision serve as the assessment measures that were employed in the comparisons. Table 1 contrasts, the KDD99 dataset performance of many machine learning tenfold cross-validation classifiers. Similarly, Table 11 compares the performance of numerous machine learning classifiers with tenfold cross-validation using the UNSW-NB15 dataset. Table 1 illustrates how well Game theory, SVM, STL-IDS, RF, RNN and CNN functionin differentiating between abnormal and regular traffic, with 99.22%, 99.83%, 99.86%, and 99.94% accuracy percentages, in that order. Using tenfold cross-validation, Game theory, SVM, STL-IDS, RF, RNN and CNNmaintain the same order of superiority utilizing the UNSW-NB15 dataset, with accuracy rates of 93.53%, 93.71%, 95.54%, and 96.07%, in that order.Tenfold cross-validation assessment comparisons of many machine learning classifiers utilizing the testing phase's supplied UNSW-NB15 and KDD99 datasets are shown. Using the data sets, an empirical examination of spam traffic classification shows that, albeit not as dramatically, DPSO performed better than Game theory, SVM, STL-IDS, RF, RNN and CNN. Its superiority over other cutting-edge algorithms, however, is more notable by a significant margin. Game theory, SVM, STL-IDS, RF, RNN and

CNNobtained accuracy rates of 95.11%, 96.01%, 96.22%, and 96.79%. As the number of trees increases, the RF classifier generally performs better, according to the result analysis from Table 1. Rather than focusing on the most important characteristic, it divides a node by selecting the top feature chosen at random from a list of features. There's a lot of diversity as a consequence, which makes the model better.

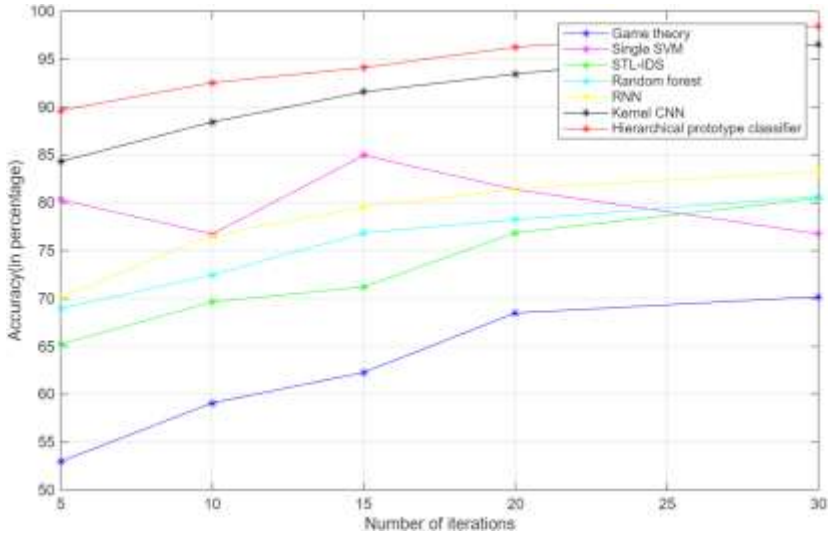

Fig 4 Accuracy comparison

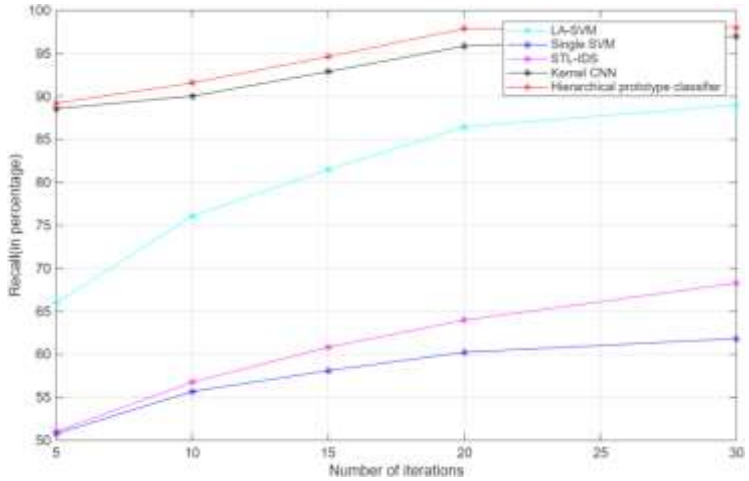

Fig 5Recall comparison

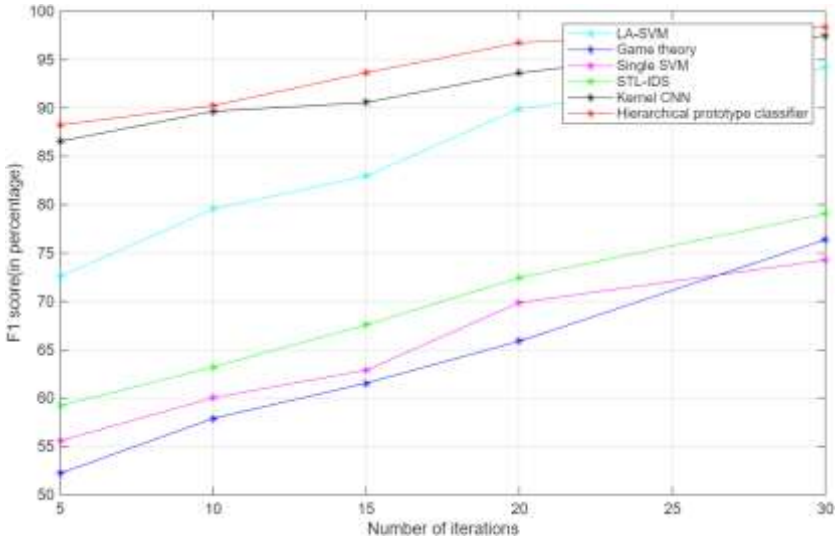

Fig 6F1-score comparison

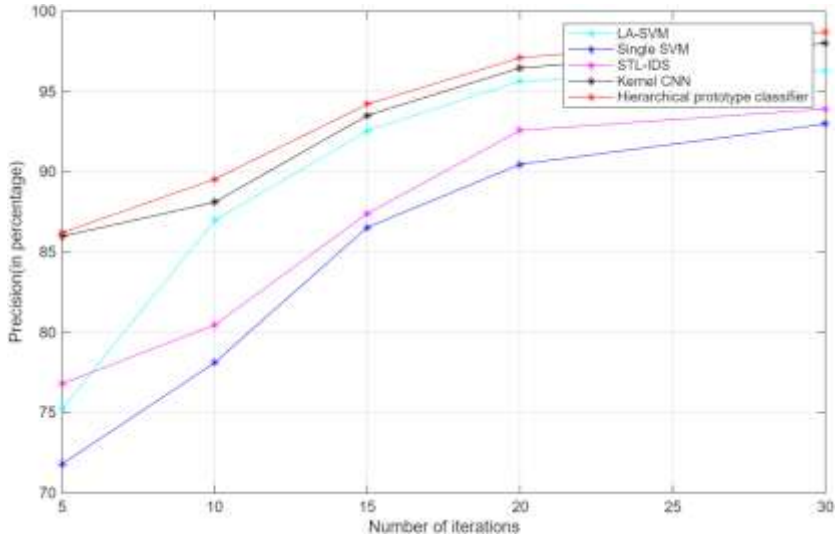

Fig 7Precision comparison

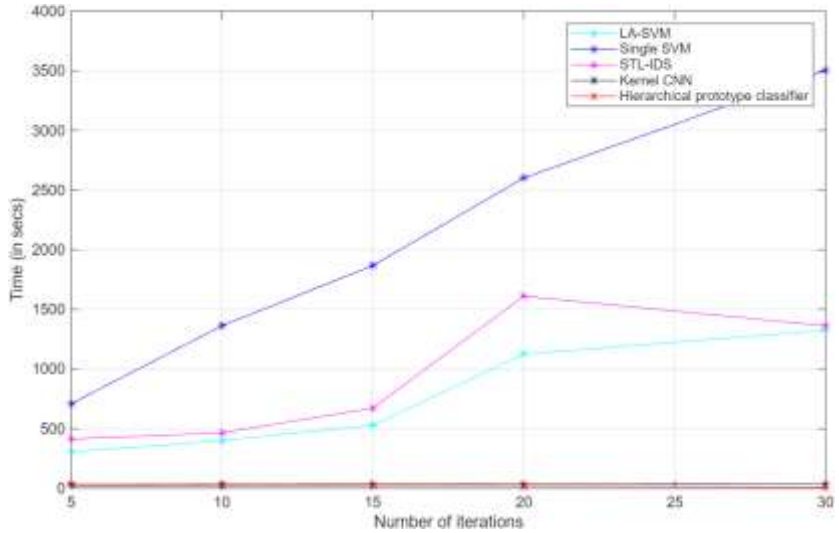

Fig 8Time comparison

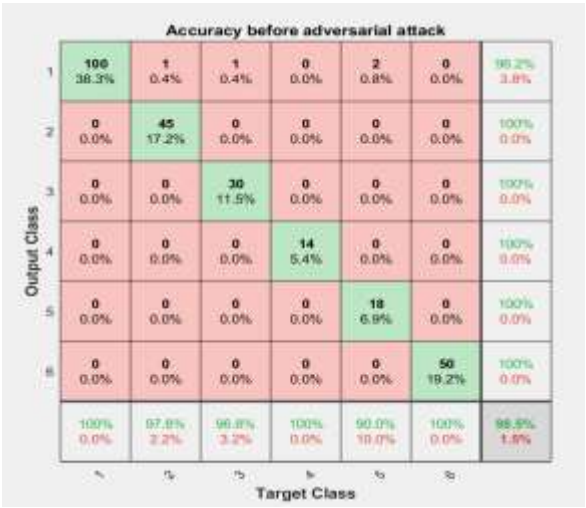

Fig 9a.Confusion matrix before attack

Fig 9b.Confusion matrix after attack

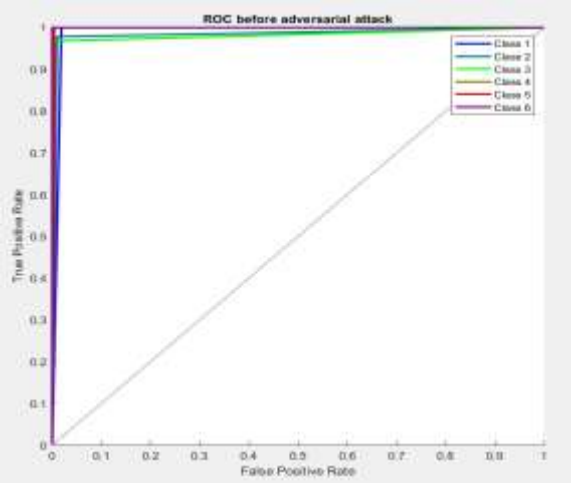

Fig 9c. ROC before attack

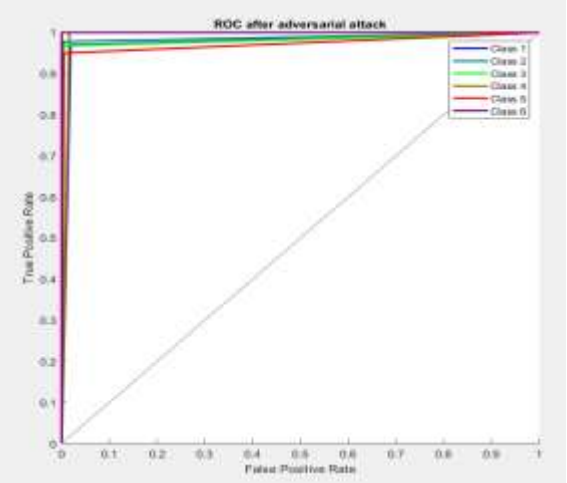

Fig 9d. ROC after attack

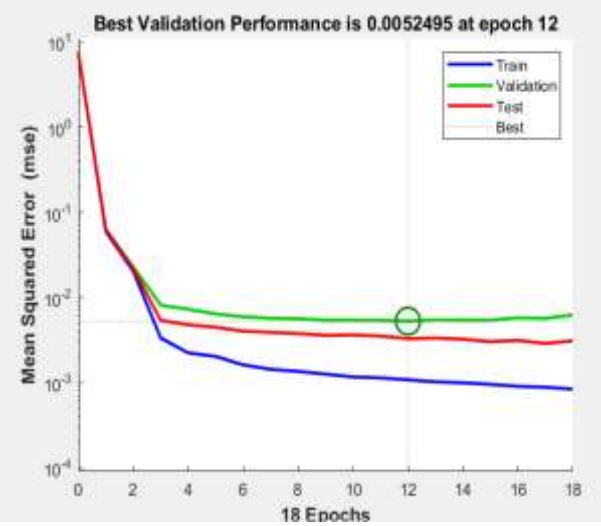

Fig 10. Epochs vs. MSE analysis

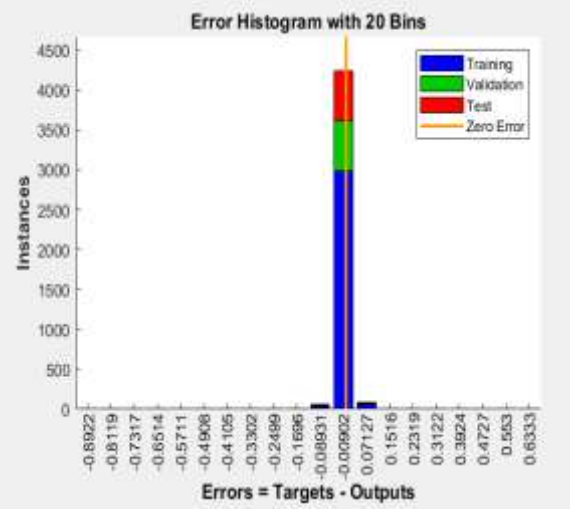

Fig 11. Error histogram

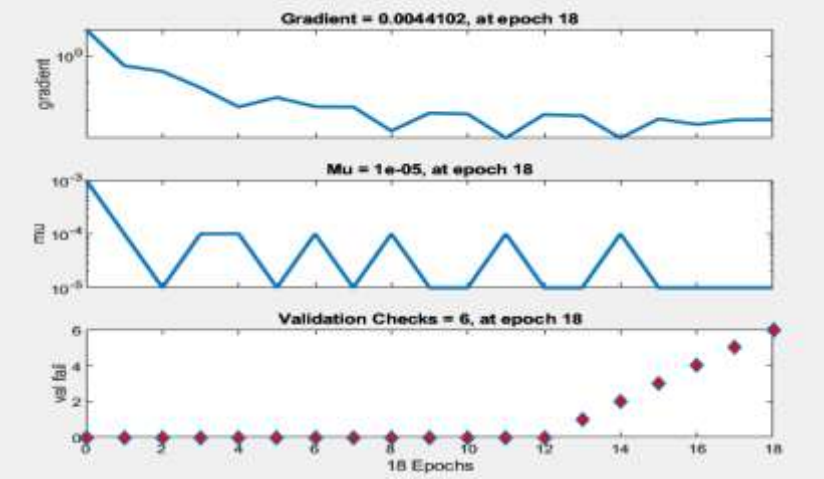

Fig 12. Validation check

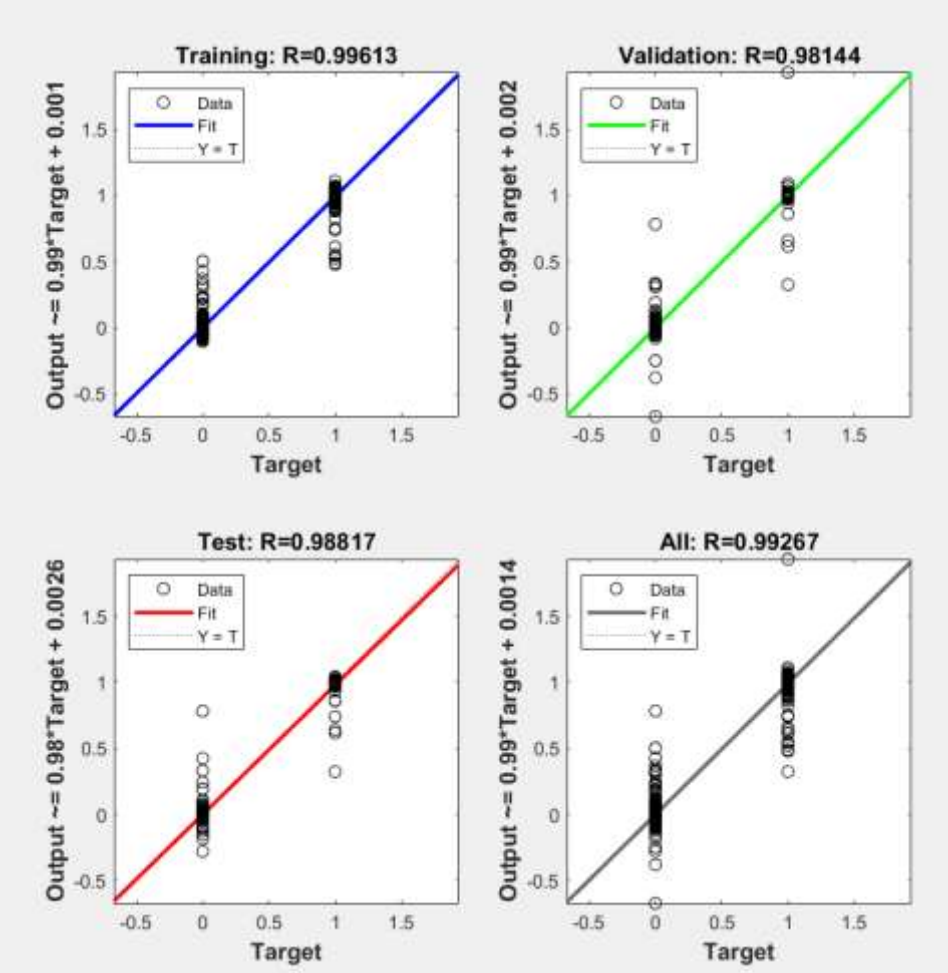

Fig 13. Training and validation analysis

Fig 10 explains the epochs and MSE analysis for training, validation and testing values. The error histogram is shown in Fig 11. Fig 12 shows the validation check and Fig 13 shows the training and validation check.Table 1 lists the time complexity of every classifier that was used for training and testing on the KDD99 and UNSW-NB15 datasets. The results indicate that because the RF classifier employs several decision trees to describe the class, it takes longer to train The comparison of accuracy across all data sets utilizing the most recent machine learning algorithms is displayed in Fig 4. Fig 5 shows that Game theory, SVM, STL-IDS, RF, RNN and CNNperform better than the other classifiers in the tenfold cross-validation test mode of both datasets; Fig 6 demonstrates that in the tenfold cross-validation test mode of both datasets, DPSO and Hierarchical classifier outperform the other classifiers; in the given test mode of both datasets, however, the proposed model outperform the other classifiers. We compared several cybersecurity-based DL models that were employed with the IDS. Using the CSE-CIC2018 dataset, Table 1 displays the accuracy and training time of different hidden nodes and learning rates in supervised and unsupervised deep learning models. The results that are shown are straight from Fig 7 and Fig 8. Using the Bot-IoT dataset, Table 1 shows the accuracy and training time of DL-supervised and unsupervised models with various hidden nodes and learning rates. The CNN outperforms both RNN and deep neural networks with an accuracy of 97.38% when the learning rate is 0.5 and there are 100 hidden nodes. Additionally, Table 1 presents the accuracy and preparation time of generative/solo models in the CSE-CIC-IDS2018 dataset utilizing various hidden node variants and learning rates. When there are 100 hidden nodes, the deep DA learns at a rate of 0.5 compared to the other three algorithms like RF, RNN and CNNachieves a higher accuracy of 97.37%.

Table 1 Overall comparison

Iterations	Accuracy	Precision	Recall	F1-measure	Time
DPSO and Hierarchical model	**98**	**99**	**98**	**98**	**100**
k-CNN (NSL-KDD)	90.1	91	96.1	95.1	650

k-CNN (UNSW_NB15)	90.5	91.5	96.5	95.6	660
k-CNN (KDD-CUP99)	91	91.5	96.6	95.4	670
LA-SVM	88.15	90.89	95.89	87.57	760
RF	64	70	72	77	1325
NB	67	73	75	79	3505
SVM	72	75	80	80	1365

Table 1 presents the accuracy and training duration of deep discriminative models with different hidden nodes and learning rates on the Bot-IoT dataset. The proposed model attains a higher level of accuracy of 98.37% and a learning rate of 0.5 with 100 hidden nodes. The proposed model has special characteristics like shared weights and local connections, which improve system efficiency and accuracy. Moreover, deep neural networks usually require less training time than other comparable methods (like RNN and CNN). In addition, Table 16 confirms the precision and training time of generative/self-learning models using different hidden nodes and learning rates from the Bot-IoT dataset. 98.39% accuracy is achieved by the deep network, this outperforms other cutting-edge algorithms. Figs. 15 and 16 display the total variation of precision for the CSE-CIC-2018 and Bot-IoT data sets, respectivelyusing interval and box plots. ML is an effective intrusion detection technique. However, there are a good number of issues with it as well that must be resolved maintaining accuracy when compressing large-scale ML models is one of the issues in deep learning. Many obsolete items in "Big data" require dependable ML models for feature learning, even if missing or noisy data is the main focus of ML models. This allows low-quality data to be prioritized for exploration. Implementing self-learning is another difficulty for ML. New attack possibilities are developing every day. Features that have been found to identify one type of assault may therefore rapidly become antiquated or insufficient for detecting other types.

Therefore, a framework that can boost accuracy, decrease calculation time, and learn features automatically is needed. In ML systems, generalization is a major difficulty. ML algorithm cannot be given a tagged sample of every issue. Consequently, in order to categorize new data, it must first be generalized using its prior samples. There isn't yet a method in ML for learning abstractions by verbal definitions. It can only function well in the absence of billions of training instances. The fact that the ML system does not understand how a neural network makes a decision or solution presents another difficulty. Even neural networks yield wonderful outcomes, but it is difficult to tell when they will fail because of a process that lacks clarity. It is not appropriate for fields like medicine where process verification is required. Another issue with ML is over-fitting the model. It alludes to an algorithm that overly accurately mimics the training set. It indicates that an algorithm has learned enough training data to cause the model's performance to suffer. Researchers can refer to the model as over-fitted or over-trained when the accuracy ceases to increase after a predetermined number of epochs. ML models demand the machines to have enough computing power, such as GPUs, to address real-world issues. These expensive and high-power processing devices are shown in Fig 4 illustrating the overall variance in accuracy for the CSE-CIC-2018 dataset. As a result, using GPUs to train data is not practical for small enterprises. Another difficulty is to shorten the GPU acceleration training period. ML models are the subject of extensive study, yet they are still unable to withstand zero-day attack.

5. CONCLUSION

This work set out to train and evaluates the ML models so that they could be applied to the binary classification task for ML-based IDS. An acceptable collection of features was sourced from the UNSW-NB 15 imbalanced datasets, and a Gini Impurity-based Weighted RF was proposed as the feature selection method. The idea behind this method was that the feature selection process may be impacted by an uneven class distribution. The datasets features were pared down using the feature selection approach known as DPSO. The following measures were used to evaluate the models' performance in order to pinpoint the incursion: FPR, F1 score, recall, accuracy, and precision. At first, every feature set from both datasets was used in the experiment. Then, using only the characteristics that had been chosen and retrieved during the feature selection and hierarchical prototype-based classifier, the experiment was conducted again. For both datasets, the model's performance with fewer features and with the entire feature space was compared. The Decision Tree performed better in both instances when the feature selection approach was applied to both datasets. Future research can develop a multiclass classification scheme for IDS that incorporates time complexity analysis, even though multiclass classification and time complexity analysis were not conducted in this work.

REFERENCES

[1] Pereira and R. M. de Moraes, ``Comparative Analysis of AODVroute recovery mechanisms in wireless ad hoc networks," IEEE LatinAmer. Trans., vol. 8, no. 4, pp. 385_393, Aug. 2010.

[2] Seo, S. Park, and J. Kim, “Improvement of Network Intrusion Detection Accuracy by Using Restricted Boltzmann Machine,” in 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), 2016, pp. 413–417.
[3] Haseeb, N. Islam, A. Almgren, and I. Ud Din, ``Intrusion preventionframework for secure routing in WSN-based mobile Internet of Things,"IEEE Access, vol. 7, pp. 185496_185505, 2019.
[4] Nur, S. Sharmin, M. A. Habib, M. A. Razzaque, and M. S. Islam,``COllaborative neighbour discovery in directional wireless sensornetworks," in Proc. IEEE Region Conf. (TENCON), Nov. 2016,pp. 1097_1100.
[5] Nur, S. Sharmin, M. A. Habib, M. A. Razzaque, and M. S. Islam,``COllaborative neighbour discovery in directional wireless sensornetworks," in Proc. IEEE Region Conf. (TENCON), Nov. 2016,pp. 1097_1100.
[6] Yan, M. Zhou, and Z. Ding, ``Recent advances in energy-ef_cient routingprotocols for wireless sensor networks: A review," IEEE Access, vol. 4,pp. 5673_5686, 2016
[7] Mandhare, V. R. School, and R. R. Manthalkar, ``QoS routing enhancementusing Metaheuristic approach in a mobile ad-hoc network," Comput.Netw., vol. 110, pp. 180_191, Dec. 2016.
[8] Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao, Y. Gao, and J. Sun,``Hypergraph-induced convolutional networks for visual classi_cation,"IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 10, pp. 2963_2972,Oct. 2019.
[9] Sherubha, “Graph Based Event Measurement for Analyzing Distributed Anomalies in Sensor Networks”, Sådhanå(Springer), 45:212, https://doi.org/10.1007/s12046-020-01451-w
[10] Sherubha, “An Efficient Network Threat Detection and Classification Method using ANP-MVPS Algorithm in Wireless Sensor Networks”, International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8 Issue-11, September 2019
[11] Sherubha, “An Efficient Intrusion Detection and Authentication Mechanism for Detecting Clone Attack in Wireless Sensor Networks”, Journal of Advanced Research in Dynamical and Control Systems (JARDCS), Volume 11, issue 5, Pg No. 55-68
[12] Piotrowski, K., Langendoerfer, P., & Peter, S. (2006). How public key cryptography incense wirelesssensor node lifetime. In SASN’06, Alexandria, Virginia, USA.
[13] Ian, F. B., Gadiel, S., & Nigel, P. S. (2005). Advances in elliptic curve cryptography. London MathematicalSociety Lecture Note Series (No. 317), Avril.
[14] Zhu, S., Setia, S., &Jajodia, S. (2003). LEAP: Efficient security mechanisms for large-scale distributedsensor networks. In Proceedings of the 10th ACM conference on computer and communicationsecurity (pp. 62–72).
[15] Panja, B., Madria, S. K., & Bhargava, B. (2006). Energy and communication efficient group key managementprotocol for hierarchical sensor networks. In SUTC’06:Proc. IEEE Int’l. Conference on sensornetworks, ubiquitous, and trustworthy comp (pp. 384–93).
[16] Zhang, X., & Wang, J. (2015). An efficient key management scheme in hierarchical wireless sensornetworks. In computing, communication and security (ICCCS), 2015 international conference(pp. 1–7).
[17] Yuan, Q., Ma, C., Zhong, X., et al. (2016). Optimization of key pre-distribution protocol based onsupernetworks theory in heterogeneous WSN. Tsinghua Science and Technology, 21(3), 333–343.
[18] Gandino, F., Ferrero, R., &Rebaudengo, M. (2017). A key distribution scheme for mobile wirelesssensor networks: q–s –Composite. IEEE Transactions on Information Forensics and Security, 12(1),34–47.
[19] Athmani, S., Bilami, A., &Boubiche, D. E. (2019). EDAK: An efficient dynamic authentication andkey management mechanism for heterogeneous WSNs. Future Generation Computer Systems, 92,789–799.
[20] Fakhrey, H., Johnston, M., Angelini, F., & Tiwari, R. (2018). The optimum design of location-dependentkey management protocol for a multiple sink WSN using a randomly selected cell reporter.IEEE Sensors Journal, 18(24), 10163–10173.
[21] Sun, B., Li, Q., &Tian, B. (2018). Local dynamic key management scheme based on layer-clustertopology in WSN. Wireless Personal Communications, 103(1), 699–714.
[22] Mawloud, O., Imene, B., Samia, A., &Bournane, A. (2018). Efficient and energy-aware key managementframework for dynamic sensor networks. Computers & Electrical Engineering, 72, 990–1005
[23] Liu, Y., & Wu, Y. (2019). A key pre-distribution scheme based on sub-regions for multi-hop wirelesssensor networks. Wireless Personal Communications, 109, 1161–1180
[24] Chu, S.-I., Huang, Y.-J., & Lin, W.-C. (2015). Authentication protocol design and low-cost keyencryption function implementation for wireless sensor networks. IEEE Systems Journal, 11(4),2718–2725.
[25] Shim, K.-A. (2017). Basis: A practical multi-user broadcast authentication scheme in wireless sensornetworks. IEEE Transactions on Information Forensics and Security, 12(7), 1545–1554.
[26] Amin, R., Islam, S. H., Biswas, G. P., &Obaidat, M. S. (2018). A robust mutual authentication protocolfor WSN with multiple base stations. Ad Hoc Networks, 75, 1–18.
[27] Wang, D., & Wang, P. (2018). Two birds with one stone: Two-factor authentication with securitybeyond conventional bound. IEEE Transactions on Dependable and Secure Computing, 15(4),708–722.
[28] Wang, D., Li, W., & Wang, P. (2018). Measuring two-factor authentication schemes for real-timedata access in industrial wireless sensor networks. IEEE Transactions on Industrial Informatics,14(9), 4081–4092.
[29] Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.
[30] Farahnakian and J. Heikkinen, “A deep auto-encoder based approach for intrusion detection system,” in 2018

20th International Conference on Advanced Communication Technology (ICACT), 2018, pp. 178–183.
[31] Ilayaraja, N. V Neeba, and C. V Jawahar, “Efficient implementation of SVM for large class problems,” in 2008 19th International Conference on Pattern Recognition, 2008, pp. 1–4.
[32] Madani and N. Vlajic, “Robustness of Deep Autoencoder in Intrusion Detection Under Adversarial Contamination,” in Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, 2018, p. 1:1--1:8.
[33] Ankit Kumar, VijayakumarVaradarajan, Abhishek Kumar, PankajDadheech, Surendra Singh Choudhary, VD Ambeth Kumar, BijayaKetanPanigrahi, Kalyana C Veluvolu,” Black hole attack detection in vehicular ad-hoc network using secure AODV routing algorithm”, Microprocessors and Microsystems, 2021
[34] VD Ambeth Kumar, S Malathi, Abhishek Kumar, Kalyana C Veluvolu, Active volume control in smart phones based on user activity and ambient noise””, sensors, 2020
[35] Abhishek Kumar, Pardeep Kumar, Ashutosh Srivastava, VD Ambeth Kumar, K Vengatesan, AchintyaSinghal, “Comparative analysis of data mining techniques to predict heart disease for diabetic patients”, Advances in Computing and Data Sciences: 4th International Conference, ICACDS 2020, Valletta, Malta, April 24–25, 2020.