IJICTR
International Journal of Information & Communication Technology Research

Volume 16- Number 4 – 2024 (20 -32)

Combination of Machine Learning Techniques Using Weighted Majority Voting for Intrusion Detection in IoT

Mohammad Hassan Nataj Solhdar*
Shohadaye Hoveizeh Campus of Technology, Shahid Chamran University of Ahvaz
Ahvaz, Iran
n.solhdar@scu.ac.ir

Nasser Erfani Majd
Shohadaye Hoveizeh Campus of Technology, Shahid Chamran University of Ahvaz
Ahvaz, Iran
n.erfanimajd@scu.ac.ir

Received: 15 April 2024 – Revised: 10 June 2024 - Accepted: 26 August 2024

*Abstract*__**The vast scale of the IoT requires robust cloud computing capabilities for data storage, management, and analysis near the network's edge. As IoT integration in business operations grows, so does the need for secure and efficient communication. Security concerns in fog and cloud environments are critical, as network attacks can severely impact IoT, fog, and cloud computing development. Intrusion detection systems (IDS) are one of the best options designed using artificial intelligence. This paper presents an IDS designed to enhance fog security against cyber-attacks using various machine learning techniques. The NSLKDD dataset was employed to develop and test the model. Performance metrics show the proposed system's superiority over existing methods. The model operates in two phases: first, a classifier ensemble of three experts processes data into binary form using five different classifiers; second, the collective output of these classifiers is merged. By using weighted majority voting (WMV), the combined output is optimized. Experimental results demonstrate that integrating opinions from multiple experts improves classifier performance across all measured criteria—Accuracy, TPR, F-measure, and FPR—proving the model's effectiveness. Specifically, the proposed method achieves a significant improvement in various metrics, with an F-measure of 94.1%, an accuracy of 91.32%, a TPR of 93.4%, and an FPR of 0.17%.**

Keywords: Internet of Things (IoT), intrusion detection, classification

Article type: Research Article

Publisher: ICT Research Institute

* Corresponding Author

I. INTRODUCTION

The Internet of Things (IoT) is an evolution of the Internet, so that the ability to connect to the Internet is given to any entity [1, 2], which is estimated to number more than 50 billion. This huge number of connected devices represents a huge amount of traffic and generated and transmitted digital data. From 10^6 to 10^{30} data is used to describe the vast amount of digital pool formed by the IoT platform. In fact, 40% of the data generated by the IoT is stored, processed, analyzed, and operated near the edges of the network where cloud deficiencies are met to meet IoT needs [3]. These shortcomings[1] and the acceleration of IoT lubricate the wheels of fog computing pattern development. On the other hand, as the depth of this digital pool increases, it becomes problematic due to different types of attacks and intrusions [4]. Based on this, various methods and techniques have been designed and implemented to protect the IoT operating system such as firewalls, data encryption and user authentication through the fog computing model. Attack and threat methods are evolving, and leave classic security techniques inefficient and ineffective to deal with IoT security To open the getway to a new generation of intrusion detection systems built using machine learning and artificial neural networks. A series of works and researches on finding the best intelligent intrusion detection system in IoT-based environments has been done for different types of applications [5, 6].

In this paper, we define an expert as a set of binary categories that together produce a binary vector of responses. An expert from an adaptive network-based fuzzy inference system (ANFIS), an expert from k-nearest neighbor (K-NN), and an expert from support vector machine (SVM) are considered on the same data set. Then a combination of the weighted majority algorithm (WMA) method is created which ensemble the opinions of three experts to reach a final decision.

In early research, it has been shown experimentally and theoretically that combination groups are more accurate than any single classification components. A combination group generated from classifiers derived from the same learning algorithm is called homogeneous, while a combination group generated from classifiers derived from different learning algorithms is called heterogeneous. For example, bagging and boosting are often used to produce homogeneous compounds, while stacking combinations can be used to produce heterogeneous compounds. The success of a combination classifier strongly depends on the diversity in the output of its component as well as the choice of method for combining these outputs into one classifier [7].

Given the massive data volumes generated by IoT devices, deep learning methods like Deep Q-learning might initially seem ideal due to their automatic feature extraction capabilities. Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been extensively applied in various domains for tasks requiring complex pattern recognition and large-scale data analysis [30,31]. However, IoT environments typically operate with constrained computational resources, making lightweight machine learning algorithms like ANFIS, SVM, and K-NN more suitable. These methods not only offer computational efficiency but also provide better interpretability and ease of deployment, critical for IoT's decentralized nature. Our method's novelty lies in its strategic combination of these algorithms, leveraging their diverse strengths through a weighted majority voting mechanism, thus ensuring real-time, scalable, and accurate intrusion detection tailored specifically for IoT networks. This approach effectively addresses IoT's unique challenges, such as limited resources and the need for low-latency responses, making it a robust security solution.

II. RELATED WORK

Because intrusion detection systems are one of the main problem-solving methods used for IoT security, there is a tendency to use more than one technique at the same time that is proposed by Alharbi et al. [8]. They provided a proof-of-concept system for IoT security implemented in the fog computing layer. The proposed system consists of a VPN server, traffic analysis engine, challenge-response unit, and firewall. Each unit thwarts certain types of attacks. The VPN server destroys the communication channels between IoT systems against sniff, spoof, and system attacks. To detect DoS and DDoS attacks, intrusion detection systems of traffic analysis units were used, in which the decision tree machine learning method was used as a classification engine.

Pajouh et al. [9] have proposed a new layer system for intrusion detection for the backbone of IoT body using a two-layer dimensional reduction engine and a two-layer classification engine. The reduce dimensions engine consisted of component analysis and linear separation analysis units, while the classification engine consisted of Naïve Bayes and CF-KNN. Naïve Bayes classification was used to classify attack records and CF-KNN classification

[1] The deficiency in this paragraph refers to the deficiency related to the cloud, which is removed by the fog

was used as the second filter layer. Using the NSL-KDD dataset [10], the proposed model achieved acceptable performance for a small number of attacks, namely the U2R and R2L classes.

Anthi et al. have proposed a predictive and adaptive intrusion detection system for IoT systems Using Wireshark software through the IoT test network for four consecutive days and using machine learning techniques on it [11]. The proposed system consists of two main stages. First, they built a real IoT lab and controlled the normal operation of each IoT device. Then, in the second stage, malicious activities were applied on these devices, which leads to anomalous network traffic. These steps feed a supervised machine learning technique with the appropriate training data that forms the core of the intrusion detection model.

Dovom et al. [12] used a fast fuzzy tree method to identify malware intrusion and classification in the Internet of Things. This type of fuzzy-based technique consists of a fuzzy top-down induction structure such as a tree, in which the nodes inside the tree are fuzzy logic calculus operators, While the leaves of these nodes are related to the fuzzy predictions applied to the input properties features. Using the Vx-Heaven dataset, their proposed model achieved high detection accuracy at a reasonable execution time.

To improve detection, Wang et al. [13] performed logarithmic density ratios to convert NSL-KDD data set features to new and better display quality features. Using the support vector machine (SVM) as the classification engine, the experimental results showed strong performance in detection rate and detection accuracy.

Zhang et al. [14] used the UNSW-NB dataset [15] using a comprehensive overview of IoT modern attack scenarios to demonstrate the effectiveness of machine learning-based intrusion detection. Although they used a simple multilayer perceptron as a classifier, they used a new feature selection engine using the Denoising Autoencoder (DAE) based on weight loss performance. This new technique focuses on the features that represented the attacks on the network.

Another application of the UNSW-NB dataset in the IoT is a forensic architecture consisting of the C4.5 decision tree, Naïve Bayes, the Association Rule Rining (ARM), and the artificial neural network (ANN). Machine learning techniques by Koroniotis et al. [16] identify and track new and complex forms of current botnet attack

As an example of the integration of SDN and IoT, Dovom et al. [17] Provide an in-depth penetration detection system for SDN-based IoT architecture that uses SDN modeling for IoT, scalability, enhancement, and flexibility purposes. While the Boltzman Machine Restricted (RBM) was used as the engine to detect intrusion. The proposed model was evaluated and validated using the KDD Cup'99 dataset and earn Performance accuracy close to 94%.

Hodo et al. [18] presented a simple multi-layered perceptron neural network trained with forwarding and backward learning algorithms to detect DoS / DDoS attacks on IoT networks. The IoT structure consists of five node sensors, one of which acts as a server amplifier node for data analysis while the others act as a client. This method was able to successfully detect DoS / DDoS attacks to 99.4% accuracy.

For use in computer networks, Mohammadi and Sabokrou [19] proposed a semi supervised intrusion detection model constructed using deep structured neural networks trained by adversarial learning. This model consists of two main stages: training and testing. The training phase, which performed using only the normal flow of NSL-KDD data set connections, consists mainly of two modules. The first module consists of an encryption-decryption network, and the second module includes a fully connected neural network, followed by the SoftMax classification. On the other Network anomalies are generated through an optimized encoder-decoder network. The test phase uses a trained neural network that results from a training phase in which KDDTest + is fully utilized. In this model, 91.39% detection accuracy was obtained by the proposed model.

A semi supervised intrusion detection system was proposed by Kumari and Varma [20] which the classification engine used a support vector machine (SVM) and Fuzzy c-means (FCM) in combination. In this model, intrusion detection performed using two classification engines: SVM and FCM. If both classifiers label an input as a normal sample, they will eventually be considered normal. However, if the input sample is labeled as an anomaly by the SVM engine and its subset is also designated by the FCM engine, it is considered an abnormal sample and the nearest circle to support higher fuzzy membership vectors as a subclass will be selected as a subgroup.

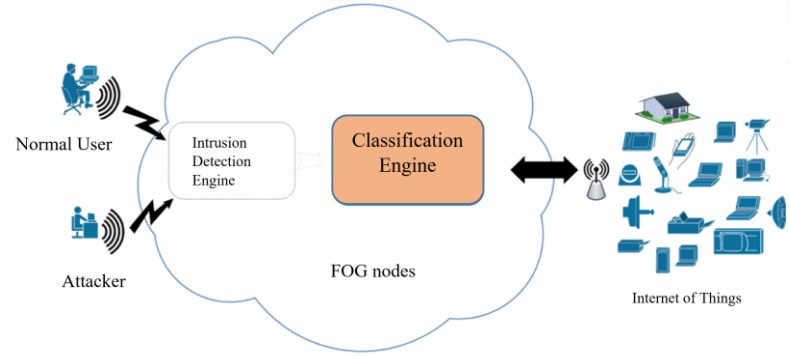

Figure 1. General framework of the proposed smart model for IoT security

Other researchers have used group training for strong IoT security. This method uses several techniques, model or expertise to solve a problem based on Artificial Intelligent. In intrusion detection, problems, group learning leads to better generalization, and voting between different group techniques offers higher detection accuracy than individual models presented by Illy et al. [21].

III. PROPOSED METHOD

In this section, the principles of architecture, concept, and design of the proposed model are presented. Figure 1 shows the general architecture of our proposed model, which is implemented in the fog calculation layer.

As shown in Figure 1, The proposed smart intrusion detection model consists of the main engine which is fully explained in Section 3.2. Traffic connections are preprocessed in the traffic processing unit, which results in the creation of traffic data in a format that is suitable for processing by the classification engine. The proposed model can be implemented in Fog calculations, which are very close to ending users and IoT devices. This model train three experts (ANFIS, SVM, K-NN) that each expert is trained by five binary classifiers to increase the predictability of attack or normal classification. The main engine shows classification-based traffic analysis, that is, network traffic that attempts to access the IoT system and analyzes security alerts in the event of detected intrusion. To clarify the roles of the expert system and the classifiers, it should be noted that in this context, the term "expert system" refers to a system based on multiple classifiers that work in combination to enhance the accuracy and reliability of intrusion detection. For this purpose, 3 votes are examined and tested, and then they are combined in a combined group. The steps are as follows, which we will explain in the following:

1- NSLKDD data preprocessing

2- Data classification with ANFIS

3- Data classification with SVM

4- Data classification with k-NN

5- Data classification with combination classifier

In our proposed intrusion detection system, we utilize a combination of ANFIS, SVM, and K-NN classifiers, each selected for its distinct advantages in data classification. ANFIS leverages the learning power of neural networks with fuzzy logic's ability to manage uncertainty, making it ideal for the complex IoT environments. SVM excels in high-dimensional data and binary classification tasks by maximizing the margin between classes, essential for distinguishing between normal and malicious traffic. K-NN is effective for handling irregular decision boundaries by classifying based on proximity to k-nearest neighbors. The integration of these methods allows the system to capitalize on their complementary strengths, thereby improving the overall accuracy and reliability of intrusion detection. This multi-classifier ensemble enhances the security of IoT networks by detecting a broader range of intrusion types.

A. Data traffic preprocessing

We used the NSL-KDD dataset [10] to train, test, and validate the model. The data attributes that represent the incoming traffic of the network system are naturally contradictory. Therefore, pre-processing of traffic data is required for the input of the classification engine [22]. The traffic preprocessor engine applies three preprocessing steps to raw traffic data: (1) Convert symbolic data to numbers. (2) Data normalization. (3) Data sampling.

1) *Convert symbolic data to numbers*

As shown in the table of attributes attached in this paper, the traffic data feature shows that after the first numeric feature, there are three symbolic fields with the titles: protocol, service, and flag features. In

this paper, we encode these properties in Table 1. These attributes are changed from symbolic to numeric:

Protocol: {tcp= 1, udp =2, icmp =3}

Private = {private =16, ..., Netsat= 30}

Flag: {pstr= 4, …, S_2= 14}

In addition, as shown in Table 1, as another step in encoding the data set, the attack subcategories are labeled as a set in their main categories, as well as the normal data class code is considered Zero. In general, we have two types of labeling: for the binary classifier engine (normal, attack), all training data records are packaged in normal and attack mode. 40 types of attacks are classified into four main categories, as shown in Table 1

2) .Data normalization

$$v' = \frac{(v - min_f)}{max_f - min_f}(max'_f - min'_f) + min'_f \quad (1)$$

In order for the range of data changes to be suitable for the input of the classifiers, the feature value of the traffic data is normal to be within a certain range. In this paper, we used the linear transformation of data with min-max formula. Assume that min_f and max_f represent the minimum and maximum values of the f^{th} feature, respectively. Therefore, the min-max formula gives the value of the f^{th} feature to the new value V in the new range.

2) *Data sampling*

This step is a fundamental step in data set preprocessing to solve the problem of data that is unbalanced in the data set. The NSL-KDD dataset contains approximately 125,000 records, of which the normal, DoS, Probe, R2L, and U2R datasets are 67343, 45927, 11656, 995 and 52, respectively. A simple calculation shows that the number of normal data contains more than 50% of the data set, and also the DoS data is large and the rest of the classes make less than 5% of the training data set. As a result, because of this imbalance, our classifiers tend to be biased toward normal, DoS data, which is more numerous, than other classes of attack, which are fewer in number.

Due to the small number of R2L and U2R attacks, the classification engine deals with these attacks as noise when training classifiers, making it difficult to distinguish this particular type of attack. One solution to this problem is to replicate both R2L and U2R attacks in different places of the data set. Repetition of this data leads to new statistics and distribution of this type of rare attack, which is shown in Table 2.

3.2 Intrusion Detection Engine

We design each classifier motor in such a way that it classifies each input into 5 binary classes. Suppose i = (1,…, 5) is an index in quintiles T = (Normal, Probe, DoS, U2R, and R2L) and Bi represents the corresponding binary classification for the target class i in T. All 5 binary classifiers are trained using the training dataset, but each classifier specifies only its own class. In other words, if the B_i classifier shows the value 1 as output, it means that this classifier specifies the input label as class i, and also if it shows the value zero as output, it means the input did not put in class i. For example, if output B_1 is 1 for an input value, that input is considered normal data, and if it is zero, that input data is not normal and will probably be one of the attacks. In order to distinguish between binary classifiers, the term expert is introduced to represent a set of 5 binary classifiers. In fact, 5 binary classifiers are called an expert. Figure 2 shows the relationship between binary classifiers and experts and presents the output format of each classifier for each class.

Experts must use a pattern to generate output, So K-NN, SVM, and ANFIS are implemented as 5 binary classifiers. Implementing in this way makes it possible to integrate and combine the experts used in a combined expert system.

In this paper, we employ Weighted Majority Voting (WMV) as the combination method for our proposed IDS. This choice is based on the method's ability to effectively balance accuracy and computational efficiency, making it particularly suitable for the resource-constrained environments typical of IoT networks. While various combination methods such as majority voting, stacking, and boosting could be considered, WMV was selected due to its proven performance in enhancing the overall decision-making process by assigning more weight to the most reliable classifiers. This ensures a robust and precise detection of different types of attacks with minimal computational overhead.

Our method is designed specifically for IoT networks, where the need for low latency and high responsiveness is paramount. By integrating WMV within the fog computing layer, close to the IoT devices, we achieve an optimized balance between security and performance, ensuring the system remains lightweight and efficient.

TABLE I. MAIN CATEGORIES AND SUBCLASSES OF ATTACKS IN THE DATA SET NSL-KDD

Subclasses	Main Categories	Numerical Code
Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm	DoS	1
Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint	Probe	2
Guess_password, Guess-passwd, ftp-write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named, Warezclient, Spy	R2L	3
Buffer-overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps	U2R	4

3.2.1 *Neuro-fuzzy classifier*

Artificial neural networks and fuzzy logic are both artificial intelligence tools that can complement each other to build another intelligent system. An artificial neural network is a low-level computational structure that works well with raw data. In contrast, fuzzy logic deals with high-level arguments obtained using the knowledge of an expert in a particular field [23]. A Neural-fuzzy network can create in two ways: combining neural network with Mamdani fuzzy or combining neural network with Sugeno fuzzy.

In 1993, Jang first introduced the neural-fuzzy adaptive inference system. It is taking into account the capabilities of fuzzy theory and artificial neural networks. Fuzzy theory is based on logical rules. Also, the artificial neural network method can extract knowledge from numerical data(Jang 1993). The system provided by Jang is called ANFIS. In the Sugeno fuzzy system, the rules are inferred as if-then. The inputs and outputs of each layer are specified and the relationship associated with it is expressed as follows [23]

Layer 1: This layer is the input layer. The nodes in this layer prepare the data for layer 2. In layer 1, no changes are made to the data so that the input is equal to the output.

Layer 2: In this layer, a fuzzy operation is performed on the data.

Layer 3: This layer is the rules layer. Each node in this layer represents a fuzzy rule. Each node in this layer receives its input from the corresponding outputs in the previous layer, and its output is the fire strength of each rule.

Layer 4: Each node normalizes its input from all nodes of the previous layer, with the output representing the normalized firing strength of the corresponding rule.

Layer 5: Each node in this layer receives its input from the corresponding node in the previous layer.

Layer 6: This layer has a single node that calculates the sum of the defuzzied values from the previous layer.

In our proposed intrusion detection system (IDS), the neuro-fuzzy classifier plays a critical role in improving detection accuracy by leveraging both ANN and fuzzy logic. The fuzzy rules in the neuro-fuzzy classifier can be generated using two methods: Grid Partition and Subtractive Clustering, which do not require expert knowledge. We use the Subtractive Clustering method to determine the number of required rules and membership functions. Then, ANFIS is employed to build the IDS.

The Subtractive Clustering method with a neighborhood radius of 0.5 (R = 0.5) is used to produce the initial fuzzy system. This method ensures that the rules and membership functions are optimized for the dataset, enhancing the system's ability to detect various types of intrusions accurately.

By detailing the layers of ANFIS and explaining how each contributes to the overall decision-making process, we demonstrate the classifier's integration within our proposed IDS. The neuro-fuzzy classifier's ability to handle both numerical data and logical rules makes it an effective component of our ensemble approach, combining its strengths with those of other classifiers to improve the system's overall performance.

3.2.2 *SVM Classifier*

Support vector machine (SVM) is an effective technique for solving classification and regression problems. SVM is essentially an implementation of the Vapnick Structural Risk Minimization (SRM) principle [25], known for its low generalization error, which is known to have a low (general) generalization error, in other words, it can be said that the training data set does not suffer from too much fit. A model is prone to overfitting or has a high generalization error if too much learning happens on the training data. SVM specifically affects data sets that are linearly separable. Therefore, the One-Versus-all method is used in data classification, which is described in Figure 2.

SVM is used in conjunction with many core functions [26]. But when the RBF kernel function is used, it produces the best results for classification [25]. This function is as follows:

$$K(x_i, x_j) = e^{\gamma \|x_i - x_j\|^2} \qquad (2)$$

TABLE II. DATA DISTRIBUTION IN STANDARD AND PROPOSED DATASETS

Data classes	Number of data in the dataset NSL-KDD	Balanced data set (suggested)
Normal	67343	33901
DoS	45926	23390
Probe	11656	5356
R2L	995	4640
U2R	52	713

The selected value for the RBF parameter is defined with the value 0.2.

3.2.3 K-NN Classifier

The nearest neighbor (K-NN) is an effective and simple technique for classifying objects based on the nearest training samples in space [27]. Consider a set of observations and objectives (x1, y1), ..., (xn, yn) in which the observations $x_i \epsilon R^d$ and the objectives $y_i \in \{0,1\}$ for ith sample that are given K-NN estimates the neighbors of a test sequence in the training sample, and uses the class labels of the nearest neighbors to predict the test class vector. Therefore, K-NN takes new samples and classifies them based on the majority of votes obtained for the k nearest sample in the training data. In K-NN, the Euclidean distance is often used as a distance measure to measure the similarity between two samples.

$$d^2(x_i, x_j) = \|x_i - x_j\|^2 = \sum_{k=1}^{d}(x_{ik} - x_{jk})2 \quad (3)$$

(x_i, x_j) R^d, $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$. Such that

The k-parameter in the K-NN classifier displays the number of neighbors in the training observation set that close to the given observations in the validation or the test data set. The difference between these parameters will affect the accuracy of each of the binary classifiers within an expert. In this research, we have considered the value of k to be 5.

3.2.4 Combined method with Weighted Majority Voting (WMV)

The idea of WMV is simple and understandable; first, the votes are assigned to each expert output. The output is accepted as the final decision with the most votes. Littlestone and Warmuth [28] showed that the number of errors made by the combination system can be reduced by introducing weights to the majority voting process.

Each expert has assigned a weight extracted from the expert's accuracy in the validation of the sample being classified. Since each expert contains 5 B_i binary classifier (as shown in Figure 2), the expert output is considered for each class i separately. Based on the output of each B_i binary classifier, expert output can be divided into two categories:

- Experts who classify the observation as an sample of class i (output value 1).
- Experts who claim that the observed sample belongs to some other class i (output value 0).

Voting procedure is repeated for each observation X and for each binary classifier within the expert. As a result, a combined expert is generated with 5 binary classifiers, for each class. This article uses the WMV method [29] to generate weight.

A set of weighting coefficients w is defined as 3 vector elements, each element j representing a weight for the jth expert in the combined group as w = (w1, w2, w3). To define a final decision function, one must first consider how the weights are used in the voting process. For a separate observation of x, 3 output values (y1, y2, y3) are obtained (one output value for each expert). Each value can be defined as a positive or negative sample as y_j= {1, - 1}. The value of 1 corresponds to the output of 1 in expert and the value of -1 indicates the output of 0 experts. The final decision y is obtained using equation (4).

$$y = \text{sgn}\left(\sum_{i=1}^{j=3} w_j \cdot y_j\right) \quad (4)$$

In this case, the value of sgn (0) is selected randomly. Each coefficient w_j is multiplied by the jth output of the most expert y_j, and the final decision is made by specifying the sum of the weight coefficients for all experts (3 experts).

The weighted majority algorithm (WMA) was first introduced in research [28] and since then, this algorithm has gained popularity as an efficient method for classification within a combination group. WMA is implemented to generate a set of weights. The WMA method uses validation data to determine the weights of each class. The WMA algorithm is as follow:

```
1: procedure WMA
2: set W←W_init
3: for i ∈ {1 . . . length(data)} do
4:         set Q_1 ←Sum(result(i, ) = =1)
5:         set Q_0 ←Sum(result(i, ) = =0)
6:         if Q_0 > Q_1 then
7:           set Q←0
8:         else if Q_0 < Q_1 then
9:           set Q←1
10:        else
11:          if rand() < 0.5 then
12:            set Q←0
13:         else
14:           set Q←1
15:        if Q = = data(i) then
16:          set W(result(i, ) ! =0)←βW(result(i, ) ! =0)
17:          set W(result(i, ) = =0)←βW(result(i, ) = =0)
```

18: return W

The pseudocode presented above pertains to the Weighted Majority Algorithm (WMA). This algorithm is used to combine the outputs of multiple classifiers to reach a final decision. Here are the explanations for the pseudocode:

First, the weights (W) are initialized using initial values.

Then, a loop is executed from 1 to the length of the data (data):

1. Calculate the number of classifiers that have an output of 1 (number of positive outputs).

2. Calculate the number of classifiers that have an output of 0 (number of negative outputs).

3. If the number of negative outputs is greater than the number of positive outputs:

- Set Q to 0 (the final decision is negative).

4. If the number of positive outputs is greater than the number of negative outputs:

- Set Q to 1 (the final decision is positive).

5. If the number of positive and negative outputs is equal:

- Generate a random number; if this number is less than 0.5:

- Set Q to 0.

- Otherwise (if the random number is not less than 0.5):

- Set Q to 1.

6. If the final decision (Q) matches the actual value of the data (data(i)):

- Reduce the weights of the classifiers that did not match the final decision by a factor of β.

- Also reduce the weights of the classifiers that matched the final decision by a factor of β.

7. Finally, return the weights.

Overall, this algorithm adjusts the weight of each classifier based on the accuracy of their outputs and ultimately produces a final decision for each input data. If multiple classifiers provide similar outputs, the algorithm assigns greater weight to those classifiers and reduces the weight of incorrect classifiers. This way, the algorithm improves and becomes more accurate over time. In this algorithm, the learning factor β is a user-defined parameter that may have values in the range $0< \beta <1$. The process is repeated for each observation in the validation sample.

IV. Evaluation Criteria and Efficiency of the Proposed Model

Various standard criteria have been proposed for evaluating intrusion detection systems. These include error detection rates and false alarm rates. The error detection rate is obtained by dividing the number of correctly detected attacks by the total number of attacks, and the false alarm rate is the ratio of the number of normal connections that are incorrectly detected as intrusions to the total number of normal connections. . The following are the evaluation criteria:

$$\text{Accuracy}=\frac{(\boldsymbol{TP+TN})}{(\boldsymbol{TP+FP+FN+TN})} \quad (5)$$

$$\text{Recall (TPR)}=\frac{(TP)}{(TP+FN)} \quad (6)$$

$$\text{Precision} = \frac{(TP)}{(TP+Fp)} \quad (7)$$

$$\text{FPR}=\frac{(\boldsymbol{FP})}{(\boldsymbol{TN+FP})} \quad (8)$$

$$\text{F\_measure} =\frac{2}{\frac{1}{Recall}+\frac{1}{Precision}} \quad (9)$$

Such that True Positive (TP): The number of intrusion samples classified as an intrusion.

True Negative (TN): The number of normal samples classified as normal.

False Positive (FP): The number of normal samples that are classified as an intrusion.

False Negative (FN): The number of intrusion samples that are normally classified.

we have examined the proposed model with the four evaluation criteria mentioned in equation 5 to 8 with the experimental data. For the three criteria Accuracy, TPR, and F_measure, the closer they are to 1, the better their performance, and the lower the False Positive Rate (FPR), the better (closer to zero).

This study was performed with MATLAB b2017 software, 64-bit installed on Windows 10 operating system, 64-bit with 7-core Intel processor, M8 up to 3.90 GHz cache, and G8 RAM. Training, validation, and experimental samples were taken from the data set presented in Table 2. In order to compare the performance of the proposed algorithms, four evaluation criteria have been considered as a basis for comparison. Here, the evaluation criteria of each binary classifier in each expert are considered separately. Experimental results for each dataset for SVM, ANFIS, k-NN and combined experts are presented separately in Tables 3 to 7.

TABLE III. ACCURACY RESULTS

Expert	Normal	DoS	Probe	U2R	R2L
ANFIS, R=0.5	96.74%	95.6651%	94.7095%	77.9773%	83.4215%
K-NN, k=5	96.53%	93.7103%	93.6149%	75.4449%	83.4124%
SVM , RBF = 0.2	96.22%	93.2904%	92.4733%	72.4296%	83.2391%
WMA	97.59%	96.2464%	95.4448%	78.794%	83.897%

TABLE IV. TPR RESULTS

Expert	Normal	DoS	Probe	U2R	R2L
ANFIS, R=0.5	95/2428	94/4822	92/8	77/9315	83/2121
K-NN, k=5	95/052	93/5433	93/84	75/342	83/1174
SVM , RBF = 0.2	94/6234	92/8336	92/276	72/1044	82/8763
WMA	96/3176	95/0116	95/1227	78/5598	83/7404

TABLE V. FPR RESULTS

Expert	Normal	DoS	Probe	U2R	R2L
ANFIS, R=0.5	0.194	0.199	0.23	0.348	0.294
K-NN, k=5	0.437	0.486	0.488	0.516	0.497
SVM , RBF = 0.2	0.629	0.689	0.685	1.3846	1.267
WMA	0.187	0.189	0.216	0.321	0.276

TABLE 6: F_MESURE RESULTS

Expert	Normal	DoS	Probe	U2R	R2L
ANFIS, R=0.5	95.43	94.51428	94. 092	77/9547	83/4015
K-NN, k=5	95.1416	93.6643	93.849	75/389	83/2248
SVM , RBF = 0.2	94.7446	92.9064	92.435	72/2574	83/0479
WMA	96.369	95.126	95.2127	78/6634	83/8186

TABLE VI. CICIDS 2017 DATASET

Expert	F_mesure	ACCURACY	TPR	FPR
ANFIS, R=0.5	93.48	90. 092	91.24	0.18
K-NN, k=5	91.43	86.57	92.31	0.34
SVM , RBF = 0.2	90.64	89.35	91.29	0.29
WMA	94.1	91.32	93.4	0.17

TABLE VII. COMPARISON WITH OTHER WORKS

Expert	F_mesure	ACCURACY	TPR	FPR
CF-KNN [9]	93.85	90. 91	92.53	0.28
Wireshark [11]	92.81	89.78	93.27	0.4
Fuzzy-Vx-Heaven [12]	93.49	88.57	92.99	0.19
Our proposed	**94.1**	**91.32**	**93.4**	**0.17**

Table 7 presents the performance results of the proposed intrusion detection model using the CICIDS 2017 dataset. This table compares the performance of different classifiers (ANFIS, K-NN, SVM) and the combined approach using Weighted Majority Voting (WMV) across four key metrics:

1. F-measure results
2. Accuracy results (two columns)
3. False Positive Rate (FPR) results

The results show:

1. ANFIS (with radius R=0.5) performs well across all metrics.
2. K-NN (with k=5) and SVM (with RBF kernel = 0.2) show competitive performance but generally lower than ANFIS.
3. The combined WMV approach consistently outperforms individual classifiers across all metrics:
    - Highest F-measure (94.1%)
    - Highest accuracy (91.32% and 93.4%)
    - Lowest false positive rate (0.17)

This table demonstrates the effectiveness of the proposed combined approach (WMV) in improving the overall performance of the intrusion detection system when applied to the CICIDS 2017 dataset. It highlights that by combining multiple classifiers, the model achieves better results than any individual classifier alone.

The inclusion of results from the CICIDS 2017 dataset also shows that the authors have extended their evaluation beyond the initially mentioned NSLKDD dataset, addressing potential concerns about the model's performance on more recent and diverse datasets.Table 8 presents a comparative analysis of our proposed method against three other established approaches in the field: CF-KNN [9], Wireshark [11], and Fuzzy-Vx-Heaven [12]. The comparison is based on four key performance metrics: F-measure, two sets of accuracy results, and False Positive Rate (FPR).

Our proposed method demonstrates superior performance across all metrics. It achieves the highest F-measure of 94.1%, surpassing CF-KNN (93.85%), Wireshark (92.81%), and Fuzzy-Vx-Heaven (93.49%). In terms of accuracy, our method outperforms the others in both sets of results, with 91.32% and 93.4% respectively. These figures represent a notable improvement over the next best performer, CF-KNN, which achieved 90.91% and 92.53%.

Particularly noteworthy is our method's False Positive Rate (FPR) of 0.17, which is the lowest among all compared approaches. This indicates that our proposed technique has the highest precision in correctly identifying positive cases while minimizing false alarms.

These results collectively suggest that our proposed method offers a more robust and accurate solution compared to existing techniques in the field. The consistent superiority across multiple performance metrics underscores the effectiveness and potential of our approach in addressing the challenges in this domain.

VI. CONCLUSION

In this paper, we present a Fog-based intrusion detection model for IoT network security. The proposed model operates in two stages. In the first stage, the classifier engine, consisting of three experts, classifies the data in binary form using five classifiers. In the second stage, the combined output from all three experts is integrated. The experimental results demonstrate that the evaluation metrics of each classifier can be enhanced by combining the opinions of different experts using the weighted majority voting (WMV) method.

To validate the effectiveness of our proposed model, we have performed extensive experiments using the NSLKDD dataset. However, to address the concerns raised, we have now extended our evaluation to include additional datasets such as CICIDS 2017. This broader evaluation allows for a more comprehensive analysis of our model's performance in different scenarios and contexts.

Furthermore, we have compared the performance of our model with several well-known intrusion detection methods, including. The comparison is based on key metrics such as Accuracy, True Positive Rate (TPR), F-measure, and False Positive Rate (FPR).

The results of these comparisons, as presented in Table 3,4,5,6 and 7, show that our proposed model consistently outperforms the other methods across all metrics. Specifically, our model achieves a higher accuracy and TPR, while maintaining a lower FPR, demonstrating its robustness and effectiveness in detecting intrusions in IoT networks.

In summary, the strategic combination of diverse machine learning classifiers through the WMV mechanism provides a significant improvement in intrusion detection accuracy and reliability. Our hybrid model is not only computationally efficient and interpretable but also scalable and flexible, making it well-suited for the dynamic and decentralized nature of IoT networks.

Future work will focus on further enhancing the model by incorporating additional machine learning and deep learning techniques, as well as exploring real-time implementation and evaluation in practical IoT environments.

REFERENCES

[1] Balasubramanian, V., et al., Low-latency vehicular edge: A vehicular infrastructure model for 5G. Simulation Modelling Practice and Theory, 2020. 98: p. 101968.

[2] Al Ridhawi, I., et al., A profitable and energy-efficient cooperative fog solution for IoT services. IEEE Transactions on Industrial Informatics, 2019. 16(5): p. 3578-3586.

[3] Mekki, K., et al., A comparative study of LPWAN technologies for large-scale IoT deployment. ICT express, 2019. 5(1): p. 1-7.

[4] Al Ridhawi, I., et al., A continuous diversified vehicular cloud service availability framework for smart cities. Computer Networks, 2018. 145: p. 207-218.

[5] da Costa, K.A., et al., Internet of Things: A survey on machine learning-based intrusion detection approaches. Computer Networks, 2019. 151: p. 147-157.

[6] 6. Quwaider, M. and Y. Jararweh, A cloud supported model for efficient community health awareness. Pervasive and Mobile Computing, 2016. 28: p. 35-50.

[7] Chen, Y., M.-L. Wong, and H. Li, Applying Ant Colony Optimization to configuring stacking ensembles for data mining. Expert systems with applications, 2014. 41(6): p. 2688-2702.

[8] Alharbi, S., et al. FOCUS: A fog computing-based security system for the Internet of Things. in 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC). 2018. IEEE.

[9] Pajouh, H.H., et al., A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. IEEE Transactions on Emerging Topics in Computing, 2016. 7(2): p. 314-323.

[10] Tavallaee, M., et al. A detailed analysis of the KDD CUP 99 data set. in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. 2009. IEEE.

[11] Anthi, E., L. Williams, and P. Burnap, Pulse: an adaptive intrusion detection for the internet of things. 2018.

[12] Dovom, E.M., et al., Fuzzy pattern tree for edge malware detection and categorization in IoT. Journal of Systems Architecture, 2019. 97: p. 1-7.

[13] Wang, H., J. Gu, and S. Wang, An effective intrusion detection framework based on SVM with feature augmentation. Knowledge-Based Systems, 2017. 136: p. 130-139.

[14] Zhang, H., et al. An effective deep learning based scheme for network intrusion detection. in 2018 24th International Conference on Pattern Recognition (ICPR). 2018. IEEE.

[15] Moustafa, N. and J. Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). in 2015 military communications and information systems conference (MilCIS). 2015. IEEE.

[16] Koroniotis, N., et al. Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. in International Conference on Mobile Networks and Management. 2017. Springer.

[17] Dawoud, A., S. Shahristani, and C. Raun, Deep learning and software-defined networks: Towards secure IoT architecture. Internet of Things, 2018. 3: p. 82-89.

[18] Hodo, E., et al. Threat analysis of IoT networks using artificial neural network intrusion detection system. in 2016 International Symposium on Networks, Computers and Communications (ISNCC). 2016. IEEE.

[19] Mohammadi, B. and M. Sabokrou. End-to-end adversarial learning for intrusion detection in computer networks. in 2019 IEEE 44th Conference on Local Computer Networks (LCN). 2019. IEEE.

[20] Kumari, V.V. and P.R.K. Varma. A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering. in 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). 2017. IEEE.

[21] Illy, P., et al. Securing fog-to-things environment using intrusion detection system based on ensemble learning. in 2019 IEEE Wireless Communications and Networking Conference (WCNC). 2019. IEEE.

[22] Aloqaily, M., et al., Data and service management in densely crowded environments: Challenges, opportunities, and recent developments. IEEE Communications Magazine, 2019. 57(4): p. 81-87.

[23] Negnevitsky, M., Artificial intelligence : a guide to intelligent systems / Michael Negnevitsky. 2005, New York: Addison-Wesley.

[24] Jang, J.-S., ANFIS: adaptive-network-based fuzzy inference system. IEEE transactions on systems, man, and cybernetics, 1993. 23(3): p. 665-685.

[25] Kuang, F., W. Xu, and S. Zhang, A novel hybrid KPCA and SVM with GA model for intrusion detection. Applied Soft Computing, 2014. 18: p. 178-184.

[26] Horng, S.-J., et al., A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Systems with Applications, 2011. 38(1): p. 306-313.

[27] Wang, W., X. Zhang, and S. Gombault, Constructing attribute weights from computer audit data for effective intrusion detection. Journal of Systems and Software, 2009. 82(12): p. 1974-1981.

[28] Littlestone, N. and M.K. Warmuth, The weighted majority algorithm. Information and computation, 1994. 108(2): p. 212-261.

[29] Dogan, A. and D. Birant. A Weighted Majority Voting Ensemble Approach for Classification. in 2019 4th International Conference on Computer Science and Engineering (UBMK). 2019. IEEE.

[30] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[31] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.

Mohammad Hassan Nataj Solhdar received his B.Sc., degree in Computer Science from the University of Mazandaran, Mazandaran, Iran in 2010, and M.Sc. degree in Computer Science from the Shahid Bahonar University of Kerman, iran in 2013. He is a faculty member of Shahid Chamran University of Ahvaz, Ahvaz, Iran. His research interests include Artificial Intelligent, Machine Learning, Neural Network, Computer Network Security, Intrusion Detection System. He has published several journal and conference papers in these fields.

Nasser Erfani Majd received his B.Sc. degree in Electronic Engineering from Shahid Chamran University of Ahvaz, Ahvaz, Iran, in 2008, and his M.Sc. degree in Electronic Engineering from Tarbiat Modares University of Tehran, Tehran, Iran in 2011. He obtained his Ph.D. degree in Electronic Engineering from Amir kabir University of Technology, Tehran, Iran, in 2016. He is currently an assistant professor in the Department of Electrical Engineering at the Shohadaye Hoveizeh Campus of Technology, Shahid Chamran university of Ahvaz, Dasht-e Azadegan, Khuzestan, Iran. His research interests include digital signal processing, delta sigma modulator-based transmitter design and mixed-signal circuit design

Attachment

Feature label	Name of network features	Descriptions	Range of values
A	Duration	Duration of connection in seconds	0-21
B	protocol_type	Type of protocol, for example, udp, tcp, etc.	Tcp – udp – icmp
C	Service	Network service at the destination, for example, http, telnet, etc.	http , talent , ftp_data , private , remote_job , mtp , eco_i ,suodup
D	Flag	Type of connection to the network	SF , S0,S1,S2,S3, REJ,SH,PSTR,…
E	src_bytes	The number of bytes of data sent from source to destination	0-1379963888
F	dst_bytes	Number of bytes sent from destination to source	0-1309937401
G	Land	1 if the connection from/to host/port is the same, otherwise zero	0-1
H	wrong_fragment	Number of wrong fragments	0-3
I	Urgent	Number of urgent packages	0-3
J	Hot	Number of host indexes	77-0
K	num_failed_logins	Number of failed logins	5-0
L	logged_in	1 if logged in; otherwise zero	1-0
M	num_compromised	Number of compromised connections	7479-0
N	root_shell	1 if root shell is obtained; otherwise zero	1-0
O	su_attempted	1 if attempts are made to implement the su root instruction, otherwise zero	1-0
P	num_root	Number of root access	7468-0
Q	num_file_creations	Number of file creation operations	43-0
R	num_shells	Number of shell command creations	2-0
S	num_access_files	Number of file access commands	9-0
T	num_outbound_cmds	Number of outbound commands for access to ftp protocol	0
U	is_host_login	1 if the input belongs to the host list, otherwise zero	1-0
V	is_guest_login	1 if the user logins as a guest; otherwise zero	1-0
W	Count	The number of connections to a host in the latest connections in the last two seconds	511-0
X	srv_count	The number of connections a single service has in more than two seconds in the last connection	511-0
Y	serror_rate	The number of connections that have SYN error	1-0
Z	srv_serror_rate	The rate of connections with SYN error (server)	1-0
AA	rerror_rate	The rate of connections that have REJ error	1-0
AB	srv_rerror_rate	The rate of connections that have REJ error (server)	1-0
AC	same_srv_rate	Percentage of connections that have the same service	1-0

AD	diff_srv_rate	Rate of connections that have different service	1-0
AE	srv_diff_host_rate	Rate of connections that have different hosts	1-0
AF	dst_host_count	The number of connections from a host to the destination during a specific time	255-0
AG	dst_host_srv_count	The number of connections from a host to destination for access to service	255-0
AH	dst_host_same_srv_rate	The rate of connections from a host to the destination to have access to the same service	1-0
AI	dst_host_diff_srv_rate	The rate of connections from a host to the destination to have access to various services	1-0
AJ	dst_host_same_src_port_rate	The rate of connections from a host to the destination from the same port	1-0
AK	dst_host_srv_diff_host_rate	The rate of connections from a host to various destinations to have access to service	1-0
AL	dst_host_serror_rate	The rate of connections from a host to a specific destination that has SYN error	1-0
AM	dst_host_srv_serror_rate	The rate of connections from a host with a specific service to a destination that has SYN error	1-0
AN	dst_host_rerror_rate	The rate of connections from a host to a specific destination that has REJ error	1-0
AO	dst_host_srv_rerror_rate	The rate of connections from a host with a specific service to a destination that has REJ error	1-0