Received December 29, 2021, accepted January 14, 2022, date of publication January 18, 2022, date of current version January 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144208

Optimized Deep Autoencoder Model for Internet of Things Intruder Detection

BADR LAHASAN[1] **AND HUSSEIN SAMMA**[1,2]
[1]Faculty of Computer and Information Technology, University of Shabwah, Shabwah, Yemen
[2]Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia (UTM), Johor Bahru, Johor 81310, Malaysia

Corresponding author: Hussein Samma (hussein.samma@utm.my)

ABSTRACT The development of an optimized deep learning intruder detection model that could be executed on IoT devices with limited hardware support has several advantages, such as the reduction of communication energy, lowering latency, and protecting data privacy. Motivated by these benefits, this research aims to design a lightweight autoencoder deep model that has a shallow architecture with a small number of input features and a few hidden neurons. To achieve this objective, an efficient two-layer optimizer is used to evolve a lightweight deep autoencoder model by performing simultaneous selection for the input features, the training instances, and the number of hidden neurons. The optimized deep model is constructed guided by both the accuracy of a K-nearest neighbor (KNN) classifier and the complexity of the autoencoder model. To evaluate the performance of the proposed optimized model, it has been applied for the N-baiot intrusion detection dataset. Reported results showed that the proposed model achieved anomaly detection accuracy of 99% with a lightweight autoencoder model with on average input features around 30 and output hidden neurons of 2 only. In addition, the proposed two-layers optimizer was able to outperform several optimizers such as Arithmetic Optimization Algorithm (AOA), Particle Swarm Optimization (PSO), and Reinforcement Learning-based Memetic Particle Swarm Optimization (RLMPSO).

INDEX TERMS Deep learning, autoencoder, IoT, anomaly detection.

I. INTRODUCTION

Recently, deep learning models showed great success for the problem of anomaly detection in IoT environment. These models include convolutional neural network (CNN) [1]–[5], long short term memory (LSTM) [6]–[8], deep autoencoders [9]–[14], deep belief neural network [15], [16], and a hybrid deep models [17]–[20].

CNN is a deep end-to-end model which is able to perform automatic feature extraction from raw input data. CNN has been utilized by Kim *et al.* [1] for Denial-of-Service attack detection in IoT networks. The basic idea of their approach is that it converts the 1D traffic features to a 2D image. Then, CNN operations are applied to encode the input 2D image, which is eventually fed to a binary classifier to classify it as an attack or normal IoT traffic. Conducted analysis of the proposed approach on CSE-CIC-IDS 2018 and KDD dataset showed that CNN is efficient in encoding traffic features, and it achieved an accuracy of 82% in the F1-score measure. An eight-layer CNN architecture model was given by Jung *et al.* [3] for IoT botnet detection in the smart health network. In their research, they focused on the pattern of power consumption to distinguish normal from abnormal IoT traffic. In their study, they collected a dataset for the consumed power of several IoT devices, including camera, Router, and Voice assistance. The power was monitored during idle time as well as during attack time. Experimental analysis showed that CNN was able to recognize malicious from normal patterns with an accuracy of 90%. A multi-CNN scheme that combines several CNN models for IoT industrial attack detection was given in [5]. Basically, they fused two CNN models and evaluated them using the NSL-KDD dataset. The results indicated that the fused scheme outperformed the single CNN model with a detection performance of 87%. The problem of unsupervised anomaly detection using CNN was discussed by Munir *et al.* [2]. They have developed a time series predictor that uses CNN to predict the next step and pass it to another deep anomaly detector to classify it as a normal or outlier pattern. The proposed approach in [2] showed a competitive performance

The associate editor coordinating the review of this manuscript and approving it for publication was Siddhartha Bhattacharyya.

in several streaming data that were used for model evaluation. A hierarchical of semi-supervised temporal convolutional network (TCN) models were studied by Cheng *et al.* [4]. They stack different TCN, and it was trained with a mix of labeled data and unlabeled instances.

LSTM models were investigated by Shi *et al.* [6], Xu *et al.* [7], and Li *et al.* [8]. In the work of Shi *et al.* [6], they have studied the effectiveness of the standard LSTM model for abnormal botnet traffic detection. Specifically, they have encoded traffic packets as a time series sequence which represents the behavior and characteristics of botnet attacks. Conducted experiments on several botnet benchmarks showed that LSTM outperformed RNN and other related models. This is due to the advantage of LSTM in handling the problem of vanishing gradients which occurs in the training of long sequences [21]. In the work of Xu *et al.* [7] an improved LSTM was introduced. The key point of their scheme is that they incorporated a time factor and a smooth activation function into LSTM to enhance its performance. To assess the improved model in [7], extensive experiments were conducted on a real dataset collected from IoT environment. Result indicated further accuracy improvements were achieved by embedding previously mentioned techniques (i.e. time factor and smooth function).

Deep autoencoder models were studied by Shone *et al.* [9], kim *et al.* [10], Gurina *et al.* [11], Telikani *et al.* [12], Lopez-Martin *et al.* [13], and Meidan *et al.* [14]. In the work of Shone *et al.* [9], they suggested a non-symmetric autoencoder model. The main concept was to evolve only the encoding phase independently without the decoding phase. Experimental analysis indicated that their model reported further accuracy improvement up to 5% against the standard autoencoder. Further work was given by kim *et al.* [10]. They have implemented a deep autoencoder model for outlier detection. Specifically, their model was built using normal IoT traffic, and then unseen traffic instances that lie outside the trained patter will be classified as suspicious behavior. The idea of flood attack detection using an autoencoder model was explored by Gurina *et al.* [11]. The designed model was used for the detection of different classes of flood attacks such as SYN flood, TCP flood, UDP flood, ICMP flood, and HTTP flood. Reported results in [11] indicated the superiority of the implemented autoencoder model in handling and classifying all mentioned flood attacks. A cost-sensitive stacked autoencoder model that comprises several hidden layers was discussed in [12]. The key point of their scheme is to set different class costs in order to balance minority/majority data. Additional study has employed a conditional variational autoencoder deep model which was applied for intruder detection in IoT network [13]. The key concept of variational autoencoders is that it depends on probability distributions to capture and encode traffic pattern.

Deep Belief Neural Network (DBNN) was applied in the work of Manimurugan *et al.* [15] for intruder detection in IoT-based smart medical environment. They have investigated several kinds of attacks such as Heartbleed, SQL injection, Infiltration, etc. In [15], optimization algorithms were employed to tune DBNN structure in terms of the number of layers and the number of neurons in each layer. Reported results indicate that DBNN achieved an F1-score of more than 97% in all kinds of attacks. Further work was introduced by Balakrishnan [16]. They applied DBNN for the prevention of various IoT network attacks, including denial of service, overflow, brute force, DNS query, cache poisoning, malware infection, and others. The reported average F1-score in [16] was 95.3%.

Hybrid deep learning models were studied by Hwang *et al.* [17], Yin *et al.* [19], and Parra *et al.* [18]. Hwang proposed a combination of CNN model with the deep autoencoder model. The hybrid model in [17] was built from a normal IoT traffic, and then DDoS pattern will be captured as an outlier. Conducted experiments indicated the successfulness of the implemented hybrid model with extremely low false-positive. Similarly, CNN with a deep autoencoder model has been employed in the work of Parra *et al.* [19] for encoding time-series sequences. In their analysis, they used Yahoo Webscope S5 time series dataset. Their hybrid model achieved an accuracy of 99%. A distributed deep learning model that combines CNN with LSTM was investigated by Parra *et al.* [18]. In their approach, LSTM was working as a backend detector on the cloud; however, CNN was deployed on IoT edge to encode attack patterns. The integrated model in [18] was evaluated using the N-baiot dataset, and it was reported of 94% F-1 score measure. The integration of deep learning with metaheuristics was studied in [20]. Basically, the Whale optimizer was integrated with LSTM to perform an automatic selection for the weights and biases. Their approach was evaluated with various benchmark datasets, including CIDDS-001, UNSWNB15, and KDD. Results in [20] showed that an accuracy above 99% was achieved in all conducted datasets. Further deep learning-based hierarchical and ensemble models were presented in [22] and [23], respectively. A table that summarized all previously discussed deep learning models is given in Table 1.

Nevertheless, deep learning models are facing the challenge of IoT resource constraints such as power, memory, and CPU support. To mitigate this challenge, optimization algorithms are considered as one promising solution [20]. However, the integration of the deep autoencoder model with optimization algorithms required a powerful optimizer that can work with high dimensions optimization. To fill this gap, this study adopts an efficient two-layers optimizer. This optimizer has several advantages, such as (i) it evolved with small population size, (ii) it has one dedicated layer for fine-tuning and one layer of exploration task, and (iii) it incorporated Q-learning to control the switching from exploration to exploitation. It should be noted that the implemented two-layers optimizer was presented in our previous research as a conference paper for the problem of large-scale optimization [24]. The main contribution of this work could be summarized in the following points:

TABLE 1. Summary of related studies.

Deep Model	Ref	Authors	Benchmarks	Accuracy	Advantages / Disadvantages
CNN	[1]	Kim et al.	They have used CSE-CIC-IDS 2018 and KDD datasets.	82% F1-score	CNN works well on 2D images, and it has the advantage of detecting spatial correlation. It is efficient in capturing and encoding local patterns. However, adopting CNN raises several challenges, such as finding the best CNN architecture, training with a large set for better performances and it required fine-tuning of its internal parameters.
	[2]	Munir et al.	They have employed Yahoo Webscope dataset, which contains 367-time series.	74% F1-score	
	[3]	Jung et al.	They have collected 8000 instances from various IoT traffic such as Botnets, Idle, etc.	90% F1-score	
	[4]	Cheng et al.	They have used the DS2OS dataset, which has 350,000 communication records captured for 11 days.	96% F1-score	
	[5]	Li et al.	They have adopted the NSL-KDD dataset.	87% accuracy	
LSTM	[6]	Shi et al.	They have used MCFP IoT botnet dataset.	98% F1-score	LSTM has the ability to capture timely dependent patterns in a long sequence. However, it requires a longer training time as compared with CNN because it processes the data sequentially.
	[7]	Xu et al.	They have evaluated the enhanced LSTM model using DS2OS dataset.	98% accuracy	
	[8]	Li et al.	They have adopted three industrial IoT datasets, including CTU-13, Gas-Water, and AWID.	95% accuracy	
Autoencoders	[9]	Shone et al.	They have used KDD Cup '99 and NSL-KDD datasets.	85% in F1-score measure	The main advantage of autoencoders is reducing the dimensionality of the features and mapping them to a lower dimension space named embedding space. However, autoencoders are prone to overfitting. As compared with PCA, autoencoders are considered computationally expensive.
	[10]	kim et al.	They have collected a dataset that contains 664,928 records from 11 July 2019 to 29 July 2019.	-	
	[11]	Gurina et al.	They have deployed MyBB web applications and performed different types of SQL injection attacks.	88% accuracy	
	[12]	Telikani et al.	They have used KDD CUP 99 and NSL-KDD datasets.	98% accuracy	
	[13]	Lopez-Martin et al.	They have employed NSL-KDD dataset.	86% F1-score	
	[14]	Meidan et al.	They have implemented N-baiot intrusion detection dataset.	99% accuracy	
Deep Belief Neural Network	[15]	Manimurugan et al.	The dataset of CICIDS 2017 was utilized.	98% accuracy	A deep belief neural network uses small labeled training data as compared with other deep models. However, it needs tuning of its topology to achieve better performance.
	[16]	Balakrishnan	They have used various benchmark datasets from the literature.	95% F1-score	
Hybrid	[17]	Hwang et al.	They have adopted USTC-TFC2016 datasets	99% accuracy	Hybridizing several models has the advantage of combining the strength of different models in a single model. However, this integration comes at the cost of increasing model complexity.
	[18]	Parra et al.	Their proposed distributed model was evaluated using N-baiot dataset.	94% F-1 score	
	[19]	Yin et al.	They have adopted Yahoo Webscope dataset.	99% F-1 score	
	[20]	Jothi et al.	They have used CIDDS-001, UNSWNB15, and KDD datasets.	87% accuracy	

- It uses a lightweight, efficient optimizer that evolved with a micro swarm (three particles only).
- It integrates the lightweight optimizer with a deep autoencoder model to enhance the accuracy and reduce the complexity (number of input features and number of hidden neurons).
- It applies the proposed optimized model for handling the problem of IoT anomaly detection, and it compares the outcomes with reported results in the literature.
- It compares the performances of the employed optimizer with other well-known and recent optimization algorithms.

The remaining part of this paper is organized as follows. The details of the proposed optimized model are explained in Section II. A series of conducted experiments implemented to evaluate the effectiveness of the proposed optimized model is shown in Section III, followed by the conclusion and future work presented in Section IV. Table 2 lists all abbreviations used in this study.

TABLE 2. List of abbreviations.

CNN	Convolutional Neural Networks
UAV	Unmanned Aerial Vehicles
ROC	Receiver Operating Characteristic
AOA	Arithmetic Optimization Algorithm
WGA	Wild Geese Algorithm
PSO	Particle Swarm Optimization
CLPSO	Comprehensive Learning Particle Swarm Optimization
RLMPSO	Reinforcement Learning-based Memetic Particle Swarm Optimization
KNN	K-nearest neighbor
LSTM	Long short term memory
IoT	Internet of Things
TCN	Temporal Convolutional Network
DBNN	Deep Belief Neural Network
Ack	Acknowledgement
Syn	Synchronize
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
TP	True Positive
TN	True Positive
FP	False Positive
FN	False Negative
PCA	Principal Component Analysis

ig.

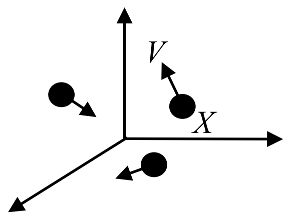

FIGURE 1. Initialization of particles X_1, X_2, X_3 in the search space.

F_1	F_2	$\ldots$	F_n	I_1	I_2	$\ldots$	I_n	M

FIGURE 2. The proposed optimization encoding scheme.

II. OPTIMIZED DEEP LEARNING AUTOENCODER

The main architecture of the proposed optimized model is given in Fig. 3. As can be seen that it contains four phases, namely, micro swarm initialization, transition using Q-learning algorithm [25], operations execution, and fitness evaluation. These phases are explained as follows.

A. MICRO SWARM INITIALIZATION

In this phase, the micro swarm population that consists of three particles is initialized with a random vector X according to the search space of the optimized problem. In addition, a velocity variable V is used with X to control the amount of jump in the search space, as shown in Fig. 1.

The length of the initialized vector X is equal to the length of the encoded problem in this study. In particular, the vector X is encoded with three parts which are IoT feature selection F, training instances selection I, and the number of autoencoder hidden neurons M as given in Fig. 2.

For IoT features selection, it has been encoded as a binary optimization problem where each bin has a variable F that could take a value of zero or one. As such, if the bin is set to one, it means the corresponding filter is selected, and it will be activated during the features extraction process. Otherwise, it will be omitted. The second part of the scheme is used to encode the training instances (malicious and normal). Therefore, for each variable, I could take a discrete value in the range of 1 to the number of instances. It is worth mentioning that half of the variables are used for selecting malicious IoT instances, and the rest are used for normal IoT instances. The length instance selection part was set to 200 with 100 for malicious and 100 for normal. The last part of the encoding scheme is used to encode the number of hidden neurons in the autoencoder. It should be noted that the variable $\boldsymbol{M}$ was configured to take a discrete value in the range of 2 to 10. In this study, it is assumed that the optimal embedding space is 2D, where it becomes easier to visualize the data, and it makes the KNN classifier works effectively.

In this is studied, the N-baiot dataset [14] is employed, which was captured by several IoT devices, namely Thermostat, a Baby monitor, a Webcam, and Doorbells. These devices were exposed to two different types of botnet attacks,

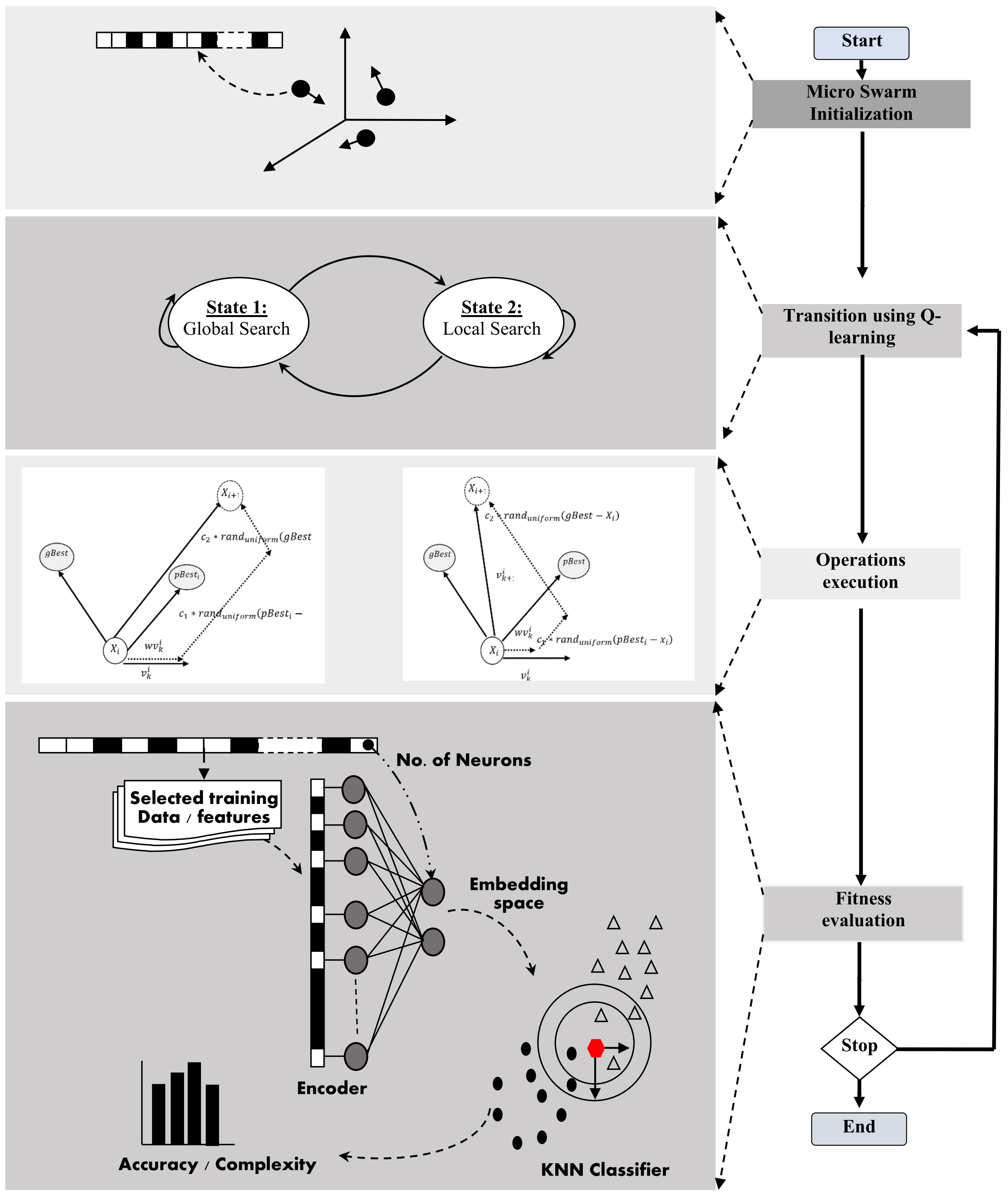
Start
Micro Swarm Initialization
State 1: Global Search
State 2: Local Search
Transition using Q-learning
X_{i+1}
gBest
$pBest_i$
$c_2 * rand_{uniform}(gBest$
$c_1 * rand_{uniform}(pBest_i -$
X_i
wv_k^i
v_k^i
$c_2 * rand_{uniform}(gBest - X_i)$
v_{k+1}^i
$c_1 * rand_{uniform}(pBest_i - x_i)$
Operations execution
No. of Neurons
Selected training Data / features
Embedding space
Encoder
Accuracy / Complexity
KNN Classifier
Fitness evaluation
Stop
End

FIGURE 3. The proposed optimized deep autoencoder model.

namely Gafgyt and Mirai as shown in Fig. 4. It should be noted that a botnet attack is a denial-of-service attack which is aiming to flood IoT networks by generating massive traffic. The N-baiot dataset has five Mirai attacks which are scanning for vulnerable devices, Acknowledgement (Ack) packets flooding, Synchronize (Syn) packets flooding, User

Datagram Protocol (UDP) packets flooding, and UDP flooding with fewer options. In addition, Gafgyt attack has five categories which are sending spam data, UDP flooding, Transmission Control Protocol (TCP) packets flooding, and sending spam data with specified Internet Protocol (IP) address and port. The complete details are given in Tables 3 and 4. A total of 115 traffic statistics were computed in N-baiot dataset. These features are related to different measures such as the mean, standard deviation, etc. The computed traffic statistics were captured from monitoring IoT traffic over different windows, namely 100 ms, 500 ms, 1.5 sec, 10 sec, and 1 min. For each window, 23 feature has been calculated from source IP, MAC address, channel. The full details can be found in [14].

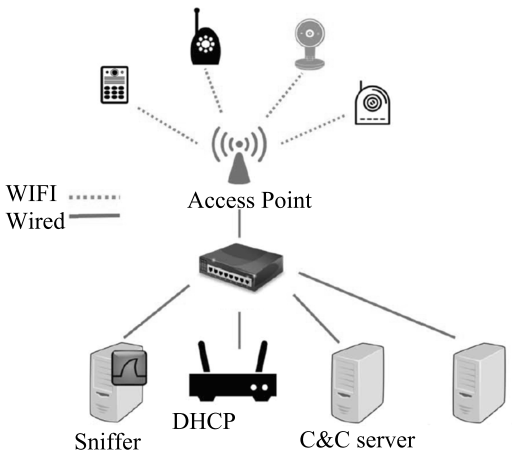

FIGURE 4. IoT attack set-up as given in N-baiot dataset [14].

B. TRANSITION USING Q-LEARNING

This layer is responsible for switching between the local search and global search modes. As described in [24], in the global search, the micro swarm particles are able to explore the search space by modifying the whole search vector; meanwhile, in the local search mode, they are allowed to modify part of the search vector (i.e., see Fig. 8). As such, there will be switching between global search and local search, which is performed under the control of the embedded Q-learning algorithm [25]. As indicated in Fig. 5, the Q-learning is modeled with two states, and a Q-table of size 2 x 2 is created to keep track of each state by rewarding each well-performing state and penalizing others (i.e., by giving a value of −1). It is worth mentioning that each particle is associated with its own Q-table, which enhances the diversity of the population and enables each particle to evolve independently from the swarm [24].

C. OPERATIONS EXECUTION

The implemented two-layers optimizer has three basic search operations, which are exploration, exploitation, and jumping search [24]. As mentioned earlier that the micro swarm has only three particles, and each particle X_i is updated based on

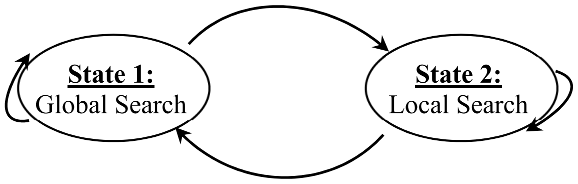

FIGURE 5. State diagram transition of Q-learning.

the following equations.

$$X_i^{t+1} = X_i^t + V_i \tag{1}$$

$$V_i = \omega * V_i + c_1 * r_1(pBest_i - X_i) + c_2 * r_2(gBest_i - X_i) \tag{2}$$

where X_i^{t+1} is the new location of currently executed particle i, V_i is particle velocity and ω is the inertia value. Parameters c_1 and c_2 are cognitive and social acceleration coefficients, respectively. Variables r_1 and r_2 are random numbers in the range (0,1). $pBest_1$ is the local best position achieved by each particle and $gBest_i$ is the global best position gained by the micro swarm. When the particle is performing an exploration search mode, it simply increases both ω (set to 0.9) and the value of c_1 (set to 2.5). At the same time, it should decrease the value of c_2 (set to 0.5), which makes it fly away from the swam, as can be seen in Fig. 6. On the other hand, exploitation search is done by flipping all these values i.e. ω is set to 0.3, c_1 set to 0.5, and c_2 set to 2.5 as shown in Fig. 7.

The jumping operation basically adds a random value X_i according to the range of the search problem, as explained in [24]. The local search operations are identical to the global search operations (i.e., exploration, exploitation, and jumping), except they are applied to update part of the search vector space, as indicated in Fig. 8.

D. FITNESS EVALUATION

The last step of the proposed optimized autoencoder model is the fitness evaluation. Mainly it is used to assess the quality of the given solution by each particle in the micro swarm. Therefore, each particle will encode a different

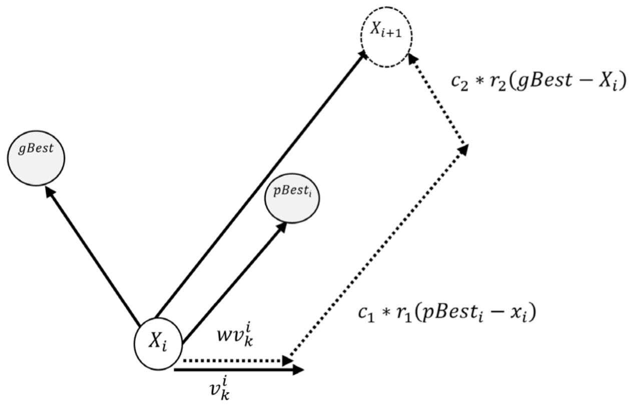

FIGURE 6. Exploration search operation.

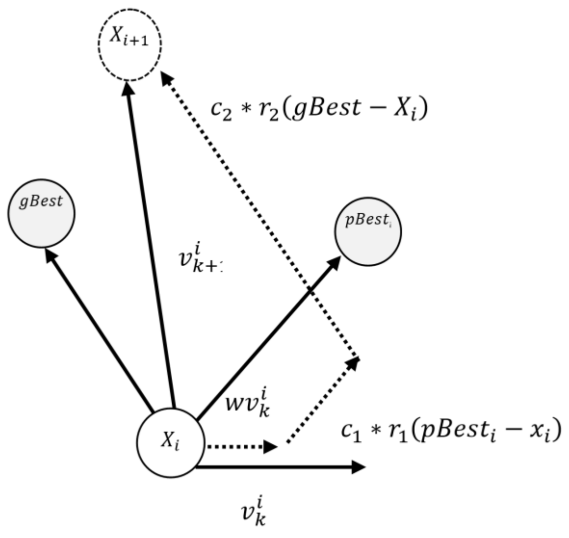

FIGURE 7. Exploitation search operation.

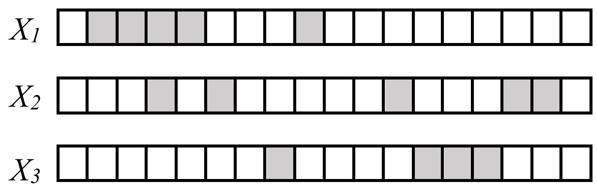

FIGURE 8. Local search mode.

solution that contains three parts, namely the selected IoT features, selected training instances, and the number of hidden neurons. According to these settings, an autoencoder model will be trained. The basic idea of autoencoder models is to reduce the dimensionality of the data by encoding the input features to a new compressed space named embedding space, as shown in Fig. 9 [26]. As can be seen, the encoding stage has input neurons that receive the input features and map them to an embedding space. The decoder stage is responsible for recovering the data back from embedding space to feature space. As such, the autoencoder will be built guided by a loss function that represents the difference between input and reconstructed data.

To minimize the computational time of fitness function, the number of training iterations of the autoencoder has been set to 100 only. Once the autoencoder has been trained, its output will be passed to a KNN classifier to classify the data and compute the accuracy rate. Here K is set to be five neighbors, as shown in Fig. 10.

To evaluate the fitness function for each particle in the micro swarm (X_i), the following formula is used.

$$fitness = -1 * (\alpha * A - \beta * C) \tag{3}$$

where A is the recognition accuracy of the KNN classifier and C is the complexity of the autoencoder. Basically, C represents the ratio of the selected features with respect to the total number of features (i.e., 115 features in N-baiot dataset). In addition, the complexity of autoencoder output neurons

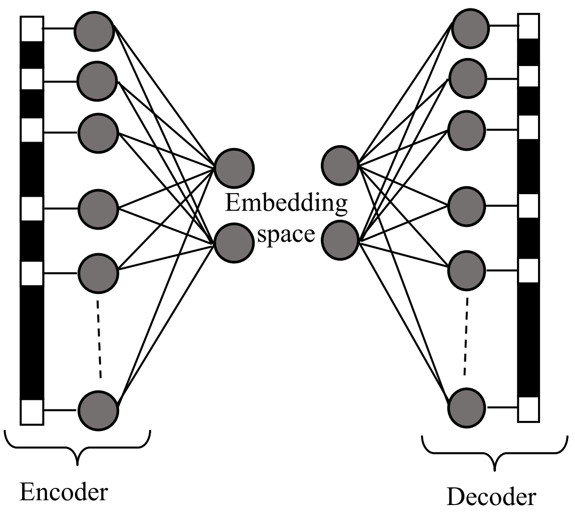

FIGURE 9. The architecture of the autoencoder model.

is computed as the ratio of the number of selected output neurons divided by 10. For illustration, the calculation of C when the number of selected features is 30, and the number of output neurons is set to 5, then C will be 0.7 (23/115 + 5/10). α and β parameters are used to control the weights and importance of A against C. Here α was set to *0.9* and β to *0.1*.

III. EXPERIMENTAL RESULTS

A. DATASET

In this study, N-baiot dataset [14] is employed for the evaluation of the proposed optimized model. The details of this dataset are given in Table 3. N-baiot dataset has 115 features, and all these features were considered in this study as in previous works [32] and [14]. These features are computed statically as the mean, the variance, the magnitude, etc., from monitoring IoT traffic over different windows, namely 100 ms, 500 ms, 1.5 sec, 10 sec, and 1 min. From each window, a total of 23 statistical features were calculated as described in Table 3.

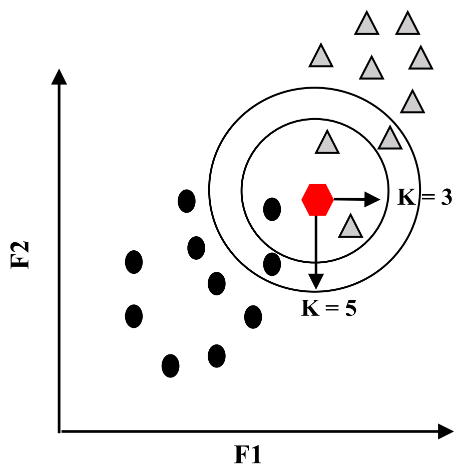

FIGURE 10. KNN classifier operation.

The dataset has been normalized where all features scaled to be in the range [0,1] using the following formula.

$$F_{o,j} = \frac{F_{i,j} - \min(F_j)}{\max(F_j) - \min(F_j)} \tag{4}$$

where $F_{o,j}$ is the output feature after normalization, $F_{i,j}$ is the input feature, $\min(F_{i,j})$ is the minimum feature value, and $\max(F_{i,j})$ is the maximum feature value.

B. PERFORMANCE MEASURES

In this study, standard evaluation measures have been used to assess the performance of the proposed optimized model. Specifically, four different measures were implemented, including the accuracy, precision, recall, and F1-score. Their mathematical formula is defined as follows.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

where *TP* is the total number of IoT instances correctly classified as malicious, *TN* is the total number of IoT instances correctly classified as normal traffic, *FP* is the total number of IoT instances wrongly classified as malicious, but they are normal traffic, and *FN* is the total number of instances wrongly classified as normal, but they are malicious traffic.

TABLE 3. N-baiot dataset features description.

Value	Statistics
Outgoing packet size	mean and variance
Packet count	number
Packet jitter	mean, variance, and number
The packet size of both outbound and inbound together	magnitude, radius, covariance, and correlation

C. PERFORMANCE ANALYSIS

This section analyzes the performances of the proposed model with a non-optimized autoencoder base model. It should be noted that the non-optimized model was trained with all input IoT features (115 feature), and the output neurons here was set to 10 neurons. In addition, the non-optimized model trained with the whole training set, i.e., 70 % of the data. Each experiment has been repeated ten times, and the mean value of all measures has been reported in Table 5. In terms of complexity, it can be seen that the optimized model uses only two hidden output neurons, and it reduced the input features to less than 36 in all studied cases (i.e., doorbell, Thermostat, baby monitor, security camera, and webcam device). In terms of accuracy, specificity, sensitivity, and F1-score, the outcomes confirmed the superiority of the optimized model in achieving better performance. This is due to the benefit of compact embedding space produced by the optimized autoencoder. This space will help the KNN classifier to work effectively due to lower dimensions.

D. AUTOENCODER EMBEDDED SPACE ANALYSIS

As a visual analysis for the outcomes of the optimized autoencoder in the embedding space. The testing data has been visualized in the optimized 2D embedding space, as can be seen in Fig. 11. As can be seen that the output of the autoencoder produces almost separable data. This will help the implemented KNN classifier correctly classify and separate the malicious traffic from the normal traffic, as indicated in Fig. 10.

E. SELECTED IoT FEATURES ANALYSIS

Further analysis has been conducted to demonstrate the most selected IoT features by the implemented two-layers optimizer over ten independent runs. Fig. 12 displayed IoT features that have been chosen in all runs. As can be seen that features are related to jitter traffic (features start with HH_jit) have been set in all the runs. This implies that jitter is one of the most valuable indicators used to distinguish malicious IoT traffic. More details about these features can be found in [14].

F. CONFUSION MATRIX ANALYSIS

The confusion matrix has been computed in this section to measure the performance of the model in terms of true positive rate (*TP*), false positive rate (*FP*), true negative rate (*TN*), and false-negative rate (*FN*). The outcome has been compared with the non-optimized model as given in Fig 13 and Fig. 14. As can be seen, the optimized model is able to eliminate most of the false alarms with a moderate cost of missing the malicious traffic (i.e., true negative rate). This is related to the advantage of generalized and compact 2D embedding space generated by the optimized autoencoder. On the other hand, the non-optimized model uses ten dimensions embedding space, as explained earlier.

G. COMPARE WITH OTHER OPTIMIZERS

This section investigates the outcomes of the proposed model as compared with other related optimizers, including PSO [27], RLMPSO [28], and AOA [29]. The settings of these algorithms are given in Table 7. Each experiment has been executed ten times, with 500 fitness evaluations given for each optimizer. The mean value of accuracy, recall, precision, and F1-score are reported in Table 6. It is clearly shown that the proposed two-layers optimizer is able to outperform other optimizers in all computed measures. One possible reason for the superiority of the two-layers optimizer is related to the advantage of working with a micro swarm population that required fewer fitness evaluations. More importantly, the implemented two-layers optimizer has the ability to switch adaptively from exploration mode to exploitation

TABLE 4. Details of N-baiot dataset [14].

Device			**Miria Attack**				
Device Model	Type	Normal	Ack	Scan	Syn	UDP	UDPPlain
Danmini	Doorbell	49,548	102,195	107,685	122,573	237,665	81,982
Ennio	Doorbell	13,113	0	0	0	0	0
Ecobee	Thermostat	39,100	113,285	43,192	116,807	151,481	87,368
B120N/10	Baby monitor	175,240	91,123	103,621	118,128	217,034	80,808
PT-737E	Security camera	62,154	60,554	96,781	65,746	156,248	56,681
PT-838	Security camera	98,514	57,997	97,096	61,851	158,608	53,785
XCS7-1002-WHT	Security camera	46,585	111,480	45,930	125,715	151,879	78,244
XCS7-1003-WHT	Security camera	19,528	107,187	43,674	122,479	157,084	84,436
SNH 1011 N	Webcam	52,150	0	0	0	0	0
Device			**Gafgyt Attack**				
Mdel	Type	Normal	Combo	Junk	Scan	TCP	UDP
Danmini	Doorbell	49,548	59,718	29,068	29,849	92,141	105,874
Ennio	Doorbell	13,113	53,014	29,797	28,120	101,536	103,933
Ecobee	Thermostat	39,100	53,012	30,312	27,494	95,021	104,791
B120N/10	Baby monitor	175,240	58,152	28,349	27,859	92,581	105,782
PT-737E	Security camera	62,154	61,380	30,898	29,297	104,510	104,011
PT-838	Security camera	98,514	57,530	29,068	28,397	89,387	104,658
XCS7-1002-WHT	Security camera	46,585	54,283	28,579	27,825	88,816	103,720
XCS7-1003-WHT	Security camera	19,528	59,398	27,413	28,572	98,075	102,980
SNH 1011 N	Webcam	52,150	58,669	28,305	27,698	97,783	110,617

TABLE 5. Comparison with non-optimized model.

Optimized Model						
Device Type	Input features	Autoencoder Neurons	Accuracy%	Recall %	Precision %	F1-score %
Doorbell	**36**	**2**	**99.75**	**99.77**	**99.11**	**99.44**
Thermostat	**28**	**2**	**99.71**	**99.77**	**99.23**	**99.49**
Baby monitor	**35**	**2**	**99.77**	**99.77**	**99.75**	**99.76**
Security camera	**31**	**2**	**99.59**	**99.71**	**98.84**	**99.28**
Webcam	**22**	**2**	**99.58**	**99.61**	**99.22**	**99.42**
Non-Optimized Model						
Doorbell	115	10	97.88	97.82	99.78	98.79
Thermostat	115	10	94.59	93.98	99.60	96.71
Baby monitor	115	10	98.85	98.80	99.10	98.95
Security camera	115	10	94.93	94.15	99.80	96.89
Webcam	115	10	95.84	95.56	99.74	97.60

at the beginning of the search process. However, PSO and AOA are time-dependent algorithms, and they start with exploration and move gradually to the exploitation mode. RLMPSO algorithm works with small population size but it requires a large number of fitness evaluations needed by the incorporated local search optimizer, as explained in [24]. This makes RLMSPO achieve the lowest results in all measures.

H. OPTIMIZERS FITNESS VALUE ANALYSIS

Fig. 15 illustrates a boxplot of the reported fitness value by each optimizer. It presents the minimum, mean, and maximum values produced by each optimizer. It can be seen that the two-layers optimizer is able to achieve the best fitness value in all conducted experiments, including doorbells, Thermostat, baby monitors, and a security camera webcam.

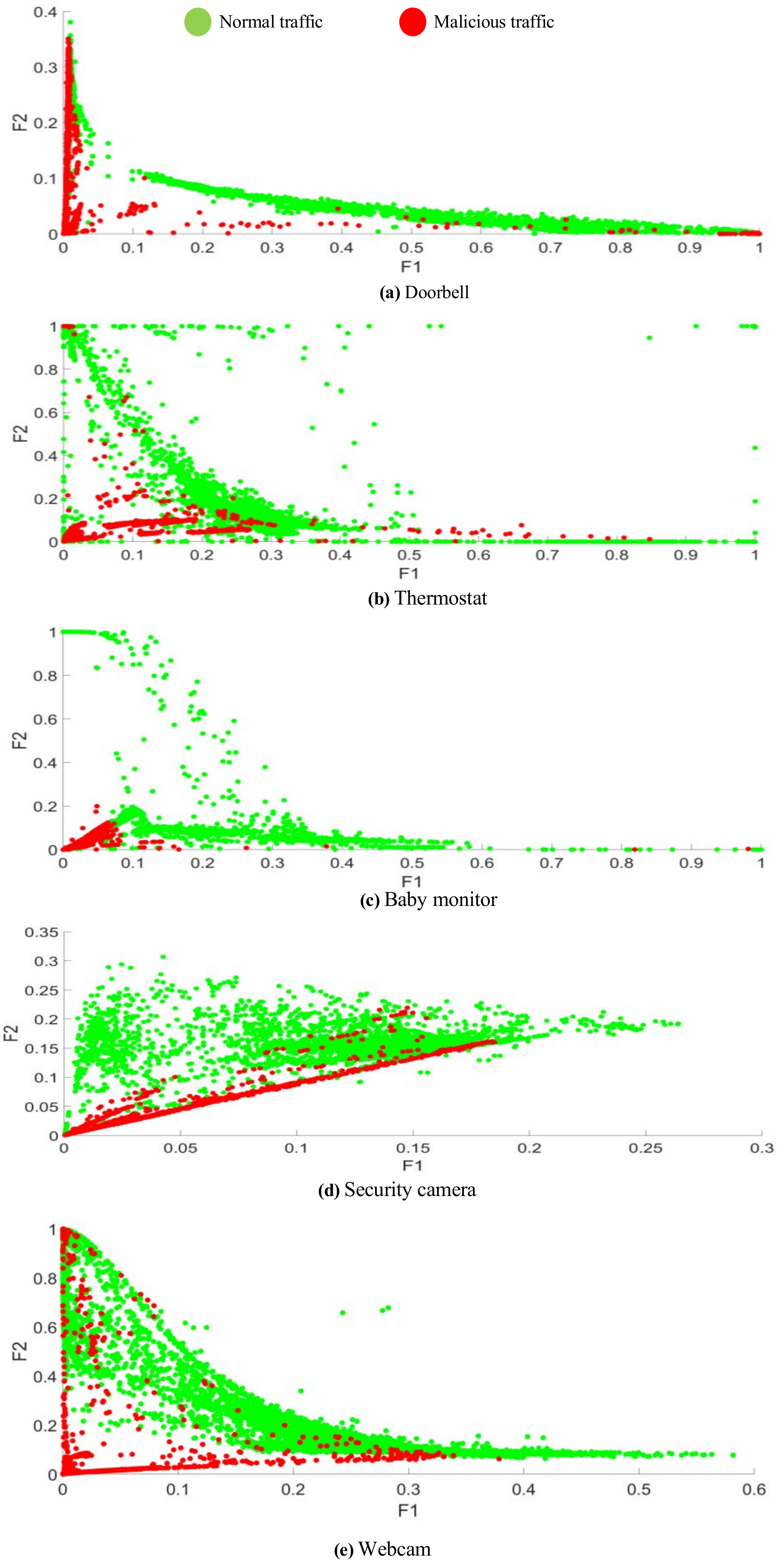

FIGURE 11. **2D embedding space data distribution.**

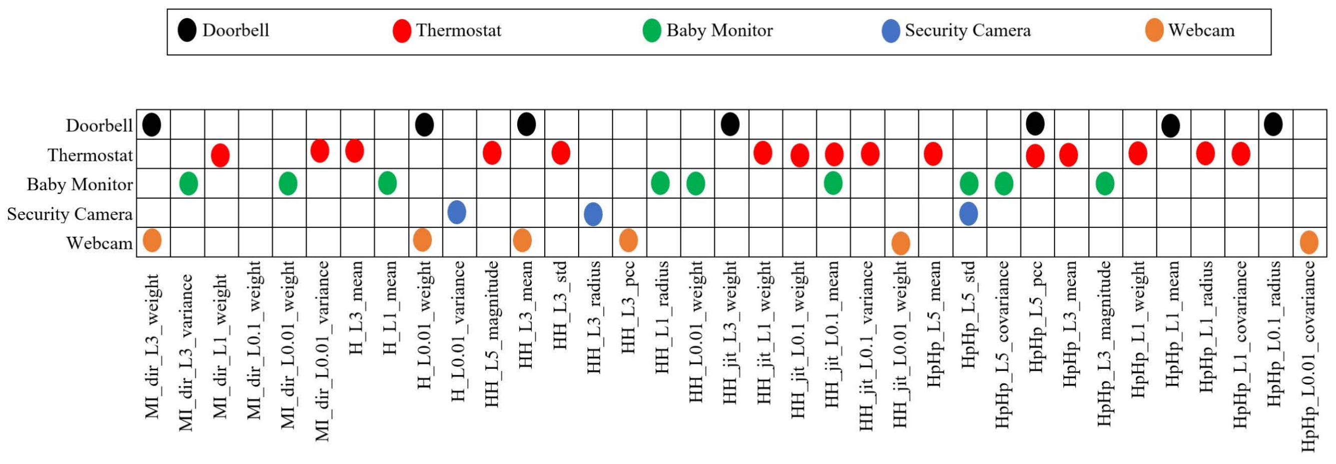

FIGURE 12. Most selected IoT features.

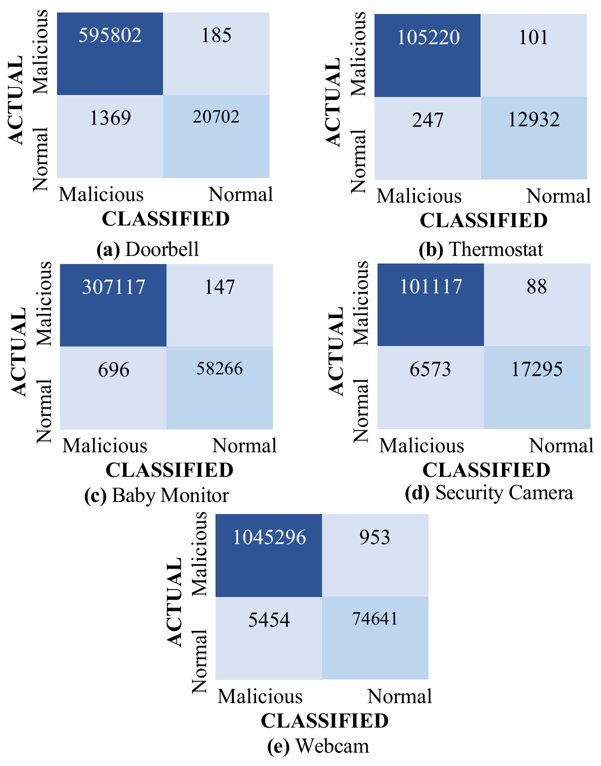

FIGURE 13. Confusion matrix of optimized model.

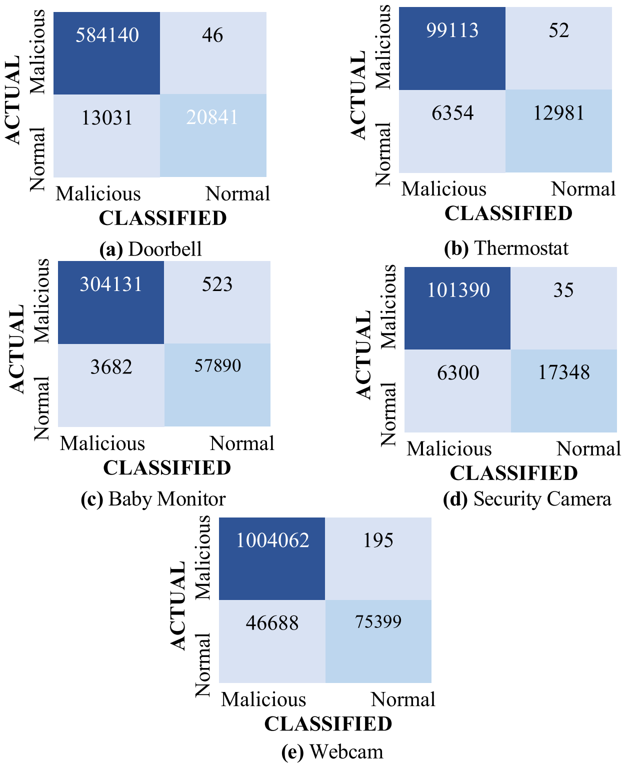

FIGURE 14. Confusion matrix of non-optimized model.

In particular, the two-layer optimizer reports a much better mean fitness value (around -0.85) with a compact boxplot. This is due to the aforementioned advantages of the implemented optimizer and also due to the dynamic transition from exploration to exploitation guided by the Q-learning algorithm.

I. STATISTICAL ANALYSIS

This section compares the outcomes of the optimized autoencoder against the non-optimized statically. Specifically, the Wilcoxon rank-sum test [31] is used where the null hypothesis H_0 assumes that the outcomes of the compared methods have the same distribution. However, the alternative hypothesis H_1 assumes the opposite. The *p-value* is set to 0.05, which means that the alternative hypothesis H_1 would be accepted when the *p-value* was less than 0.005 (95% confidence level). The results of this test are given in Table 8. It is clearly shown that the optimized model significantly outperformed the non-optimized model in all measures with a *p-value* less than 0.05. This is owing to the benefits of the generated 2D embedding space, which helps the KNN classifier to classify the data correctly.

TABLE 6. Comparison with other optimizers.

Device Type	Input features	Autoencoder Neurons	Accuracy %	Recall %	Precision %	F1-score %
			Two-Layers Optimizer [24]			
Doorbell	**36**	**2**	**99.75**	**99.77**	**99.11**	**99.44**
Thermostat	**28**	**2**	**99.71**	**99.77**	**99.23**	**99.49**
Baby monitor	**35**	**2**	**99.77**	**99.77**	**99.75**	**99.76**
Security camera	**31**	**2**	**99.59**	**99.71**	**98.84**	**99.28**
Webcam	**22**	**2**	**99.58**	**99.61**	**99.22**	**99.42**
			RLMPSO [28]			
Doorbell	47	2	98.96	98.95	99.28	99.12
Thermostat	41	2	98.61	98.66	98.18	98.42
Baby monitor	35	2	99.57	99.61	99.33	99.47
Security camera	37	2	94.67	93.90	99.49	96.61
Webcam	32	2	99.43	99.48	98.74	99.11
			PSO [27]			
Doorbell	42	2	99.75	99.77	99.11	99.44
Thermostat	41	2	98.63	98.66	98.33	98.50
Baby monitor	32	2	99.77	99.77	99.75	99.76
Security camera	36	2	99.60	99.72	98.85	99.28
Webcam	30	2	99.43	99.48	98.74	99.11
			AOA [29]			
Doorbell	43	2	99.12	99.11	99.34	99.22
Thermostat	41	2	98.61	98.66	98.18	98.42
Baby monitor	28	2	99.62	99.64	99.47	99.55
Security camera	35	2	99.59	99.71	98.84	99.28
Webcam	28	2	99.35	99.37	99.11	99.24

Furthermore, the obtained fitness value by the proposed two-layers optimizer has been compared statically with other optimizers, which are RLMSPO, PSO, and AOA. The results are shown in Table 9, and it is indicated that the *p-value* of the Wilcoxon rank-sum test is less than 0.05 in all compared optimizers. This implies that the two-layers optimizer significantly outperformed other optimizers in terms of reported fitness value.

J. COMPUTATIONAL TIME ANALYSIS

In this section, the computational time of the proposed autoencoder model has been computed and compared with a non-optimized model. As explained earlier, the non-optimized model was trained with an input feature of 115 and the output neurons set to 10. The hardware and software specifications are given in Table 12. The required time by both optimized and non-optimized autoencoder is given in Table 10. It is clearly shown that an optimized autoencoder can reduce the computational time by 33% as compared with the non-optimized model. Specifically, it needs only one microsecond to recognize one IoT input instance; however, the non-optimized model needs around 1.5 microseconds. This confirms the benefits of the implemented two-layers optimizer in reducing the complexity of the autoencoder model. Additional computational time analysis has been conducted by computing the required training time in each stage presented in Figure 3. In particular, the time needed for the initialization of the swarm, transition, search operations, and fitness evaluation is computed and depicted in Table 11 as can be seen that most computational time is consumed in the fitness evaluation step. This is due to the required time to train and evaluate both the autoencoder model and

TABLE 7. Algorithm's parameter settings.

Optimizer	Settings
PSO [27]	Population size 30, c1=1.4 and c2= 1.4, w:0.9 - 0.4
AOA [29]	Population size 30, MOP: 0.2-1, Alpha = 5, Mu=0.499
RLMPSO [28]	Population size 3, c1=2.5 in exploration mode, c1=0.5 in exploitation mode, c2=0.5 in exploration mode, c2=2.5 in exploitation mode, and w=0.9 in exploration mode, w-0.4 in exploitation mode.
Two-Layers [24]	Population size 3, c1=2.5 in exploration mode, c1=0.5 in exploitation mode, c2=0.5 in exploration mode, c2=2.5 in exploitation mode, and w=0.9 in exploration mode, w-0.4 in exploitation mode.

TABLE 8. Wilcoxon rank-sum of the optimized model against non-optimized model ($p < 0.05$).

	Accuracy	Recall	Precision	F1-score
p-value	0.0302	0.0302	0.0302	0.0302

KNN classifier. Other steps need a negligible amount of time as shown in Table 11.

K. COMPARE WITH THE LITERATURE

This section compares the outcomes of the proposed model with other reported results in the literature that employed N-baiot dataset. In particular, the F1-score achieved in work given by Al Shorman *et al.* [32] is reported Table 13. The presented results showed that the proposed optimized model achieved a better F1-score in all case studies, i.e., doorbell, Thermostat, baby monitor, security camera, and webcam. One reason is due to the advantage of using an autoencoder model to map the features to a compact and separable embedding space. It is worth mentioning that in [32], they fed the selected features by their optimizer directly to the one-class SVM classifier. In this case, the once-class required a lot of computation time to find the optimal decision boundary. In contrast, this study utilizes both the ability of a two-layers optimizer to reduce the input features and the benefits of the autoencoder model to map the selected features to a lower-dimensional embedding space (2D). This will result in speeding up the recognition time of KNN as well as enhancing its generalization due to the reduction in model complexity.

L. MODEL EVALUATION USING IoTID20 DATASET

To further validate the effectiveness of the proposed approach, the IoTID20 [33] intruder detection dataset has been used in this section. IoTID20 is a public dataset, and it has 66 features. The data has been divided into 70% to 30% for training, testing respectively. The outcome of the proposed approach is compared with other deep learning models, namely 1D-CNN and LSTM. These models were selected because they can work directly on 1D sequence patterns (i.e., IoTID20 features).

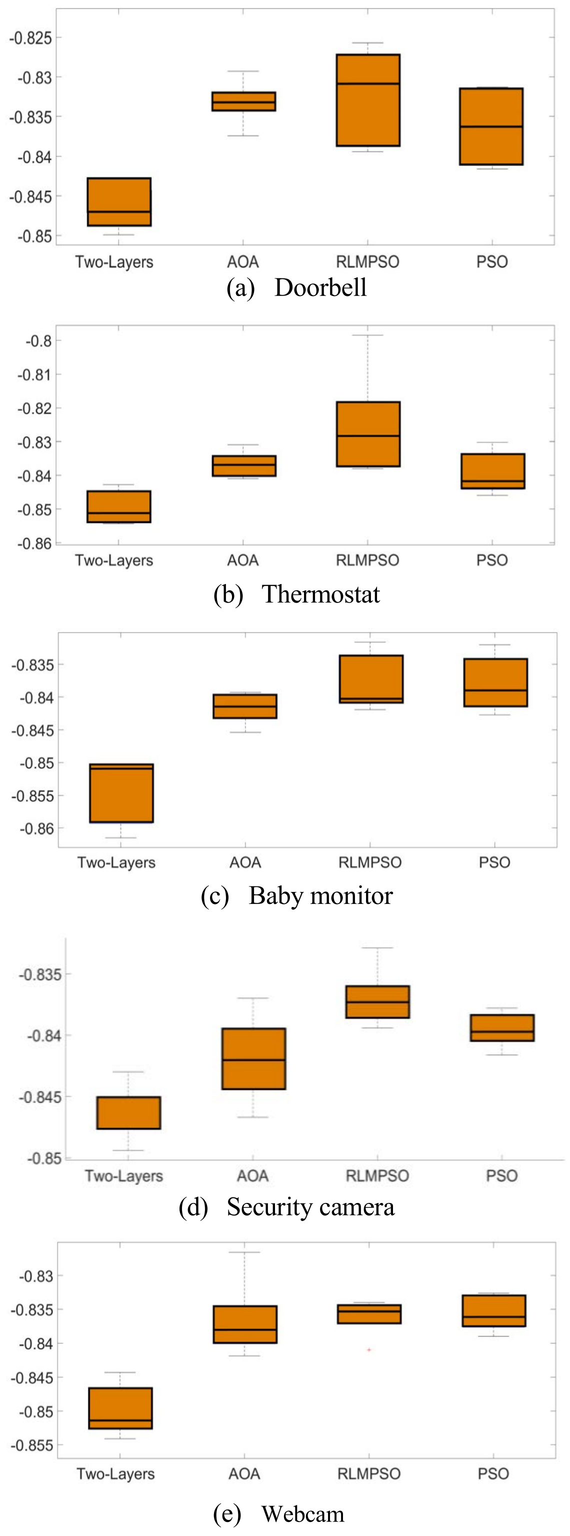

FIGURE 15. Reported fitness value by optimizers.

TABLE 9. Wilcoxon rank-sum of the proposed two-layers optimizer against other optimizers ($p < 0.05$).

	AOA[29]	RLMPSO[28]	PSO[27]
p-value	0.0396	0.0009	0.0291

TABLE 10. Computational time analysis.

	Non-optimized model	Optimized model
Time (microsecond)	1.5	**1**

TABLE 11. Training time analysis.

Step	Time (sec)
Initialization	0.000012
Transition	0.00005
Search Operations	0.000024
Fitness computation	3

TABLE 12. The details settings of the system.

Item	Settings
CPU	i7-8700
Frequency	3.2 GHz
RAM	32 GB
Hard drive	512 GB SSD
Operating system	windows 10
Language	MATLAB 2021a

TABLE 13. Compare with the work in [32].

Device Type	Al Shorman et al. [32]	Optimized model
Doorbell	95.60	**99.44**
Thermostat	97.50	**99.49**
Baby monitor	98.70	**99.76**
Security camera	94.10	**99.28**
Webcam	96.80	**99.42**

The accuracy measure of the conducted analysis is shown in Table 14, and it can be seen that all models almost report the same results to some extent. Nevertheless, the proposed approach has the advantage of working with small input features where the two-layers optimizer was able to reduce the features up to 62%. This is resulted in a shallow, deep model compared with 1D-CNN and LSTM.

TABLE 14. Performance results o IoTID20 dataset.

	Accuracy %	# of features
The proposed approach	**98.12%**	**25**
1D-CNN	97.86%	66
LSTM	97.79%	66

IV. CONCLUSION, LIMITATION AND FUTURE DIRECTIONS

This work introduced a novel optimized deep learning-based autoencoder model applied for the problem of anomaly detection in IoT networks. Basically, The optimized model was constructed using an efficient two-layers optimizer that works with a micro swarm population, i.e., three particles. Specifically, The two-layers optimizer performed simultaneous IoT features selection, training instances selection, and autoencoder neurons selection. The formulated fitness function that guided the two-layers optimizer was the accuracy of the KNN classifier that takes the output of the autoencoder as well as the complexity of the autoencoder model. The experimental results on N-baiot dataset confirmed the superiority of the proposed optimized model as compared with the non-optimized model. Moreover, the implemented two-layers optimizer achieved the best results in terms of fitness value as compared with other well-known optimizers, including PSO, RLMPSO, and AOA. Statically, the non-parametric Wilcoxon rank-sum statistical test confirms the significance of the obtained results.

Nevertheless, the proposed model needs further improvements to minimize the number of IoT input features. This will further reduce the complexity of the autoencoder model and make it able to work in a real-time IoT environment. This could be done by the development of a heterogeneous optimizer that works cooperatively as a single model. Further ideas that could be investigated in the future are validating the model using other benchmark datasets and extending the model to work for multiclass attacks recognition. Another future research avenue that could be investigated is the application of the proposed optimizer for the fine-tuning of explainable artificial intelligence presented in [38].

REFERENCES

[1] J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, Jun. 2020.

[2] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2018.

[3] W. Jung, H. Zhao, M. Sun, and G. Zhou, "IoT botnet detection via power consumption modeling," *Smart Health*, vol. 15, Mar. 2020, Art. no. 100103.

[4] Y. Cheng, Y. Xu, H. Zhong, and Y. Liu, "Leveraging semisupervised hierarchical stacking temporal convolutional network for anomaly detection in IoT communication," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 144–155, Jan. 2021.

[5] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, ''Robust detection for network intrusion of industrial IoT based on multi-CNN fusion,'' *Measurement*, vol. 154, Mar. 2020, Art. no. 107450.

[6] W.-C. Shi and H.-M. Sun, ''DeepBot: A time-based botnet detection with deep learning,'' *Soft Comput.*, vol. 24, no. 21, pp. 16605–16616, Nov. 2020, doi: 10.1007/s00500-020-04963-z.

[7] R. Xu, Y. Cheng, Z. Liu, Y. Xie, and Y. Yang, ''Improved long short-term memory based anomaly detection with concept drift adaptive method for supporting IoT services,'' *Future Gener. Comput. Syst.*, vol. 112, pp. 228–242, Nov. 2020.

[8] X. Li, M. Xu, P. Vijayakumar, N. Kumar, and X. Liu, ''Detection of low-frequency and multi-stage attacks in industrial Internet of Things,'' *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8820–8831, Aug. 2020.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, ''A deep learning approach to network intrusion detection,'' *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[10] S. Kim, C. Hwang, and T. Lee, ''Anomaly based unknown intrusion detection in endpoint environments,'' *Electronics*, vol. 9, no. 6, p. 1022, Jun. 2020.

[11] A. Gurina and V. Eliseev, ''Anomaly-based method for detecting multiple classes of network attacks,'' *Information*, vol. 10, no. 3, p. 84, 2019.

[12] A. Telikani and A. H. Gandomi, ''Cost-sensitive stacked auto-encoders for intrusion detection in the Internet of Things,'' *Internet Things*, vol. 14, Jun. 2021, Art. no. 100122.

[13] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, ''Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT,'' *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[14] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, ''N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders,'' *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul./Sep. 2018.

[15] S. Manimurugan, S. Al-Mutairi, M. M. Aborokbah, N. Chilamkurti, S. Ganesan, and R. Patan, ''Effective attack detection in internet of medical things smart environment using a deep belief neural network,'' *IEEE Access*, vol. 8, pp. 77396–77404, 2020.

[16] N. Balakrishnan, A. Rajendran, D. Pelusi, and V. Ponnusamy, ''Deep belief network enhanced intrusion detection system to prevent security breach in the Internet of Things,'' *Internet Things*, vol. 14, Jun. 2021, Art. no. 100112.

[17] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, ''An unsupervised deep learning model for early network traffic anomaly detection,'' *IEEE Access*, vol. 8, pp. 30387–30399, 2020.

[18] G. De La Torre Parra, P. Rad, K.-K.-R. Choo, and N. Beebe, ''Detecting Internet of Things attacks using distributed deep learning,'' *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102662.

[19] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, ''Anomaly detection based on convolutional recurrent autoencoder for IoT time series,'' *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 1, pp. 112–122, Jan. 2022.

[20] B. Jothi and M. Pushpalatha, ''WILS-TRS—A novel optimized deep learning based intrusion detection framework for IoT networks,'' *Pers. Ubiquitous Comput.*, vol. 21, pp. 1–17, Jun. 2021, doi: 10.1007/s00779-021-01578-5.

[21] A. Graves, ''Long short-term memory,'' in *Supervised Sequence Labelling With Recurrent Neural Networks*. Berlin, Germany: Springer, 2012, pp. 37–45.

[22] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, ''A hierarchical hybrid intrusion detection approach in IoT scenarios,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–7.

[23] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, ''Kitsune: An ensemble of autoencoders for online network intrusion detection,'' 2018, *arXiv:1802.09089*.

[24] H. Samma, S. A. Suandi, and J. Mohamad-Saleh, ''Two-layers particle swarm optimizer,'' in *Proc. IEEE Int. Conf. Autom. Control Intell. Syst. (I2CACIS)*, Jun. 2020, pp. 165–169.

[25] C. J. C. H. Watkins and P. Dayan, ''*Q*-learning,'' *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[26] G. E. Hinton and R. R. Salakhutdinov, ''Reducing the dimensionality of data with neural networks,'' *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[27] J. Kennedy and R. Eberhart, ''Particle swarm optimization,'' in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, 1995, pp. 1942–1948.

[28] H. Samma, C. P. Lim, and J. Mohamad Saleh, ''A new reinforcement learning-based memetic particle swarm optimizer,'' *Appl. Soft Comput.*, vol. 43, pp. 276–297, Jun. 2016, doi: 10.1016/j.asoc.2016.01.006.

[29] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, ''The arithmetic optimization algorithm,'' *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113609.

[30] S. Mirjalili, ''SCA: A sine cosine algorithm for solving optimization problems,'' *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016, doi: 10.1016/j.knosys.2015.12.022.

[31] T. P. Hettmansperger and J. W. McKean, *Robust Nonparametric Statistical Methods*. Boca Raton, FL, USA: CRC Press, 2010.

[32] A. Al Shorman, H. Faris, and I. Aljarah, ''Unsupervised intelligent system based on one class support vector machine and grey wolf optimization for IoT botnet detection,'' *J. Ambient Intell. Hum. Comput.*, vol. 11, no. 7, pp. 2809–2825, Jul. 2020.

[33] I. Ullah and Q. H. Mahmoud, ''A scheme for generating a dataset for anomalous activity detection in IoT networks,'' in *Proc. Can. Conf. AI*, 2020, pp. 508–520.

[34] S. Dorafshan, R. J. Thomas, and M. Maguire, ''SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks,'' *Data Brief*, vol. 21, pp. 1664–1668, Dec. 2018.

[35] P. Miao and T. Srimahachota, ''Cost-effective system for detection and quantification of concrete surface cracks by combination of convolutional neural network and image processing techniques,'' *Construct. Building Mater.*, vol. 293, Jul. 2021, Art. no. 123549.

[36] M. Ghasemi, A. Rahimnejad, R. Hemmati, E. Akbari, and S. A. Gadsden, ''Wild Geese algorithm: A novel algorithm for large scale optimization based on the natural life and death of wild geese,'' *Array*, vol. 11, Sep. 2021, Art. no. 100074.

[37] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, ''Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,'' *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[38] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, ''XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures,'' *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4225–4246, Dec. 2021.

BADR LAHASAN received the Ph.D. degree in computer science (computer vision and image analysis) from Universiti Sains Malaysia, in 2017. He is currently an Assistant Professor with the Faculty of Computer and Information Technology, University of Shabwah, Shabwah, Yemen. His research interests include face recognition, biometrics, pattern recognition, and artificial intelligence.

HUSSEIN SAMMA received the B.Eng. degree in computer engineering from Yarmouk University, Jordan, in 2006, the M.Eng. degree in computer engineering from the Jordan University of Science and Technology, in 2009, and the Ph.D. degree in computer vision and machine learning from Universiti Sains Malaysia, in 2016. He is currently an Assistant Professor with the Faculty of Computer and Information Technology, University of Shabwah, Shabwah, Yemen. He is also working as a Senior Lecturer with the Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia. His research interests include computer vision, deep learning, swarm intelligence, and soft biometrics. He is a member of IEEE Computational Intelligence Society. He has served as a reviewer for several well-known international journals, such as machine vision and applications.