Power System Protection and Control
ISSN:1674-3415

MRFO-MOENET: A MIXTURE OF EXPERTS TRANSFER LEARNING MODEL OPTIMIZED BY MANTA RAY FORAGING ALGORITHM FOR SMART IOT INTRUSION DETECTION

Farqad Abdullah Mohammed[1], Saba Joudaki[2], Mezher H Mezher[3], Mahdi Mosleh[4]

[1]Department of computer engineering, Isf.C., Islamic Azad University, Isfahan, Iran
[2]Department of computer engineering, Khor.C., Islamic Azad University, Khorramabad, Iran
[3]Department of Electronics and Communication, College of Engineering, University of Al-Qadisiyah,Al-Diwaniyah, Iraq
[4]Department of computer engineering, Isf.C., Islamic Azad University, Isfahan, Iran
Corespondance Author: Saba Joudaki
saba.joudaki@iau.ac.ir

Received: 27/12/2025 **Revised**: 26/01/2026 **Accepted**: 22/02/2026

***ABSTRACT*:**

With the rapid growth of the Internet of Things (IoT), ensuring the security of connected devices has become increasingly critical due to the rising frequency of sophisticated cyber-attacks. This paper presents MRFO-MoENet, a novel hybrid approach that combines deep transfer learning with a Mixture of Experts (MoE) ensemble architecture, optimized using the Manta Ray Foraging Optimization (MRFO) algorithm, to enhance IoT attack detection. The proposed framework leverages multiple pre-trained deep learning models, transferring knowledge from large-scale domains to IoT-specific data, and employs a dynamic expert selection mechanism for ensemble prediction. MRFO is utilized to fine-tune the parameters of both individual experts and the gating network, enabling optimal collaboration among models. Experiments conducted on the Edge-IIoTset dataset demonstrate that MRFO-MoENet achieves a high detection accuracy of 99.92% while maintaining a very low false alarm rate. These results confirm the robustness and practical applicability of our method for securing IoT environments against modern cyber threats.

***Keywords**: Internet of Things (IoT), intrusion detection, transfer learning, Mixture of Experts, Manta Ray Foraging Optimization (MRFO), metaheuristic optimization, Edge-IIoTset.*

INTRODUCTION

The exponential growth of Internet of Things (IoT) technologies across domains such as healthcare, smart homes, transportation, and industrial automation has brought forth new levels of efficiency and connectivity. However, this rapid expansion has also exposed IoT environments to diverse and sophisticated cyber-attacks. Due to hardware constraints like limited memory, low computational capacity, and energy inefficiency, IoT devices often lack built-in robust security mechanisms, making them highly vulnerable to threats such as unauthorized access, data tampering, DDoS attacks, and malware infections [1].

Traditional intrusion detection systems (IDS), particularly signature-based models, are not well-suited for these environments. They struggle with the detection of new and previously unseen attack types, and require frequent manual updates and comprehensive attack signature databases [2]. Although machine learning and deep learning-based IDS solutions have emerged to overcome some of these limitations, they also present challenges. These include dependency on large labeled datasets, high computational demand, and often poor generalization when applied to dynamic and heterogeneous IoT scenarios [3], [4].

To mitigate these limitations, recent research has emphasized the use of **deep transfer learning (DTL)** approaches, where models pre-trained on large-scale datasets are fine-tuned on IoT-specific attack data. This approach enhances generalizability and reduces the need for extensive labeled data [5]. Moreover, the use of **ensemble learning** techniques, particularly expert-based mechanisms like **Mixture of Experts (MoE),** has proven effective in improving robustness and detection performance by aggregating the predictions of multiple specialized models.

DOI: 10.46121/pspc.54.1.24

In this study, we introduce **MRFO-MoENet**, a novel detection architecture that integrates **deep transfer learning** with an **optimized Mixture of Experts ensemble**, guided by the **Manta Ray Foraging Optimization (MRFO)** algorithm. Unlike conventional ensemble approaches that apply static model aggregation, MoENet dynamically weighs expert contributions using a gating network, allowing the framework to adapt more effectively to varying types of IoT traffic. MRFO, inspired by the intelligent foraging behavior of manta rays, serves as a meta-heuristic algorithm to optimize hyperparameters of the experts and the gating strategy for better convergence and performance [6].

The proposed method is evaluated using the **Edge-IIoTset** benchmark dataset, which contains a diverse range of IoT attack scenarios. The results demonstrate substantial improvements in detection accuracy, false alarm rate, and computational efficiency compared to existing methods.

The remainder of this paper is organized as follows: Section 2 reviews related work in the field of IoT attack detection. Section 3 describes the proposed MRFO-MoENet framework. Section 4 presents the experimental setup, results, and evaluation. Finally, Section 5 concludes the paper and outlines future research directions.

RELATED WORKS

The detection of network intrusions within Internet of Things (IoT) ecosystems has attracted considerable attention in recent years, with researchers employing a range of machine learning (ML) and deep learning (DL) strategies. This section highlights notable contributions by focusing on the techniques applied, datasets utilized, and their empirical outcomes.

In [7], Prathapchandran and Janani (2021) presented a trust-aware lightweight intrusion detection approach, RFTrust, aimed at countering sinkhole attacks in RPL-based IoT infrastructures. Their method, which integrates Random Forest (RF) with Semantic Logic (SL), enhances routing reliability by isolating compromised nodes. Performance evaluation metrics such as delivery ratio, throughput, latency, and energy usage confirmed its superior efficiency when benchmarked against existing protocols.

Sharma et al. [8] (2021) examined ML-based security solutions in IoT environments enabled by 5G. Their study assessed the strengths and weaknesses of ML and DL methods, advocating for their integration with emerging technologies such as blockchain, edge and fog computing to address evolving cybersecurity threats and facilitate the development of intelligent IoT infrastructures.

Tseng, Chou, and Chao [9] (2011) proposed an intrusion detection algorithm based on metaheuristic swarm intelligence. Drawing inspiration from bird flocking behavior, the algorithm relies on pheromone trail intensities to detect anomalies. Nodes with pheromone levels surpassing a predefined threshold are flagged as potential intrusion points.

Lima Filho et al. [10] (2019) developed a hybrid detection system by combining genetic algorithms, decision trees, and neural networks. Their five-layer neural network structure was optimized through genetic algorithms, enhancing input relevance and classification performance for anomaly detection tasks.

Zong and Huang [11] (2021) conducted a comprehensive meta-analysis of intrusion detection systems (IDS) using data mining. They reviewed numerous evaluation methodologies, offering insights into the effectiveness of different IDS architectures in varied IoT settings.

Radhika and Kulothungan [12] (2019) undertook a comparative analysis of various credit scoring models, including logistic regression, LDA, k-NN, SVM, and neural networks across eight real-world datasets. Their findings revealed that SVM and neural networks consistently achieved superior performance in classification accuracy.

In [13], Bouke et al. (2023) designed an intelligent detection model for DDoS attacks utilizing decision trees and a Gini index-based feature selection mechanism. Applied on the UNSW-NB15 dataset comprising over 1.1 million samples, the model attained an accuracy of 98%. The feature selection strategy not only enhanced detection performance but also mitigated the risk of overfitting by reducing feature dimensionality.

DOI: 10.46121/pspc.54.1.24

Mustafa et al. [14] (2023) proposed a detection system based on Long Short-Term Memory (LSTM) networks, tailored for identifying adversarial DDoS attacks. Their system effectively learned long-term temporal patterns and achieved accuracy levels between 91.75% and 100% when tested against GAN-generated traffic.

In [15], Khanday et al. (2023) developed a lightweight IDS using data preprocessing and a combination of traditional ML and DL classifiers. They applied the framework on BOT-IoT and TON-IoT datasets to classify DDoS traffic, noting significant class imbalance in BOT-IoT compared to TON-IoT, which influenced classification performance.

Rani et al. [16] (2023) investigated the detection and mitigation of SYN and Slowloris attacks within Device-to-Device (D2D) communication scenarios. Using a simulated attack environment and a bespoke dataset, they demonstrated the capability of their ML model to accurately identify and thwart ongoing DDoS threats.

Lastly, Hasan et al. [17] (2023) focused on securing smart grid communication channels by identifying DDoS vulnerabilities and presenting a hybrid detection method grounded in machine learning. Their solution aimed to bolster system resilience and ensure uninterrupted operation in the face of complex cyberattacks.

PROPOSED METHOD

The architecture of the proposed method, **MRFO-MoENet**, is illustrated in Figure 1. This model is a hybrid deep learning framework designed for intelligent intrusion detection in IoT environments. It integrates multiple transfer learning models in a Mixture of Experts (MoE) structure and optimizes their performance using the Manta Ray Foraging Optimization (MRFO) algorithm. The method is composed of the following key phases:

1. Data Preprocessing and Feature Engineering: In the initial step, raw IoT traffic data is normalized using Min-Max scaling to ensure uniform feature distribution. Feature extraction techniques, including statistical and deep representations, are applied to enhance data representation. Finally, the data is partitioned into training, validation, and testing subsets to ensure robust model evaluation.
2. Construction of Expert Models Using Transfer Learning: Multiple pretrained deep learning models (e.g., ResNet152V2, InceptionResNetV2, and Xception) are fine-tuned on the IoT dataset. These models act as experts within the Mixture of Experts framework. Their intermediate feature representations are extracted and fused for enriched learning.
3. Mixture of Experts Aggregation: The outputs of the expert models are aggregated through a gated mechanism that assigns dynamic weights to each expert based on input features. This adaptive combination allows the model to leverage the strengths of each expert for different types of intrusion patterns.
4. Optimization of Hyperparameters Using MRFO: The Manta Ray Foraging Optimization (MRFO) algorithm is employed to optimize key hyperparameters of the expert models (e.g., learning rate, batch size, number of fine-tuned layers) and the gating mechanism. MRFO's adaptive foraging behavior enables efficient exploration of the hyperparameter space, leading to improved classification performance.
5. Meta-classification Layer: The final decision is made using a high-level **meta-classifier**, such as a Random Forest or SVM, trained on the combined feature outputs from the expert models. This layer enhances generalization by mitigating overfitting and capturing complex decision boundaries in the feature space.

DOI: 10.46121/pspc.54.1.24

Power System Protection and Control
ISSN:1674-3415

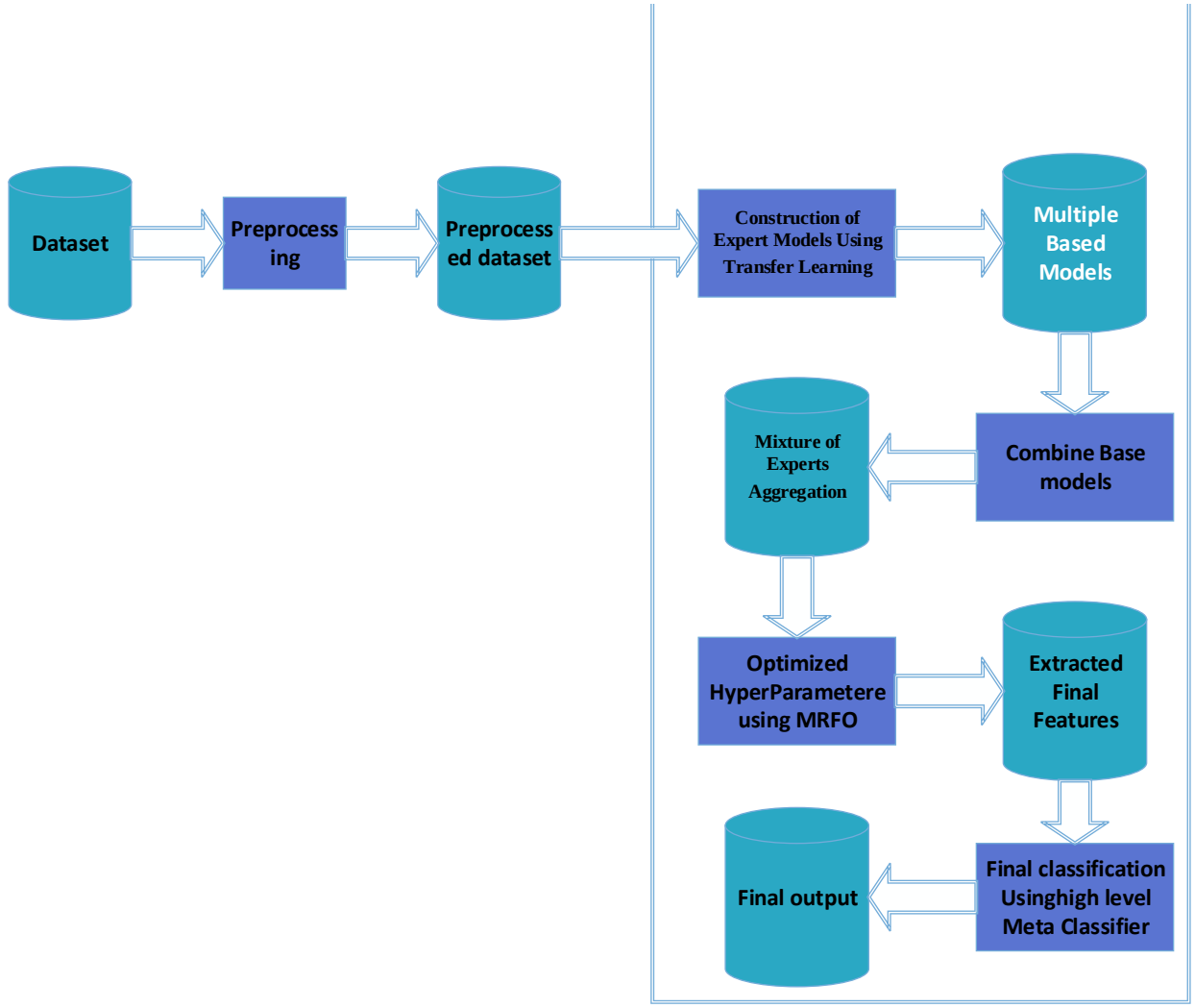

Fig. 1. Overall block diagram of proposed method

The overall pipeline ensures high detection accuracy, robustness to imbalanced datasets, and adaptability to various IoT-based attack types.

3.1 Pre-processing

In this section, pre-processing is done on the primary raw data. Preprocessing plays a very important role in data analysis. The three basic steps here for data preprocessing include data cleaning, data transformation and normalization, which are shown in the figure 2.

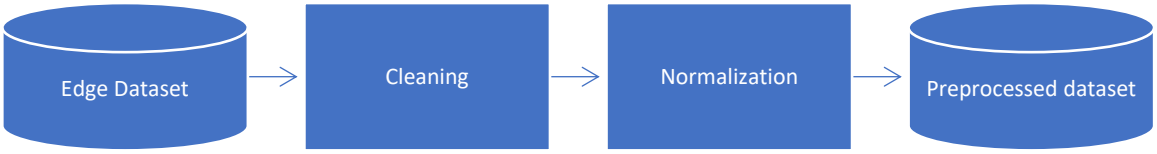

Fig. 2. Preprocessing procedure

The features are normalized and outliers are removed. For the normalization of numerical data, the minimum-maximum normalization equation is used:

(1)

As it is known in equation (1), is the old data before normalization, is the maximum value of feature x and is the minimum value of feature x, and is the new value.

DOI: 10.46121/pspc.54.1.24

Power System Protection and Control
ISSN:1674-3415

3.2 The Proposed MRFO-MoENet framework

This section introduces the core architecture of the proposed MRFO-MoENet framework, an optimized ensemble classifier that integrates transfer deep learning (DTL) with meta-heuristic optimization, as illustrated in Figure 3.

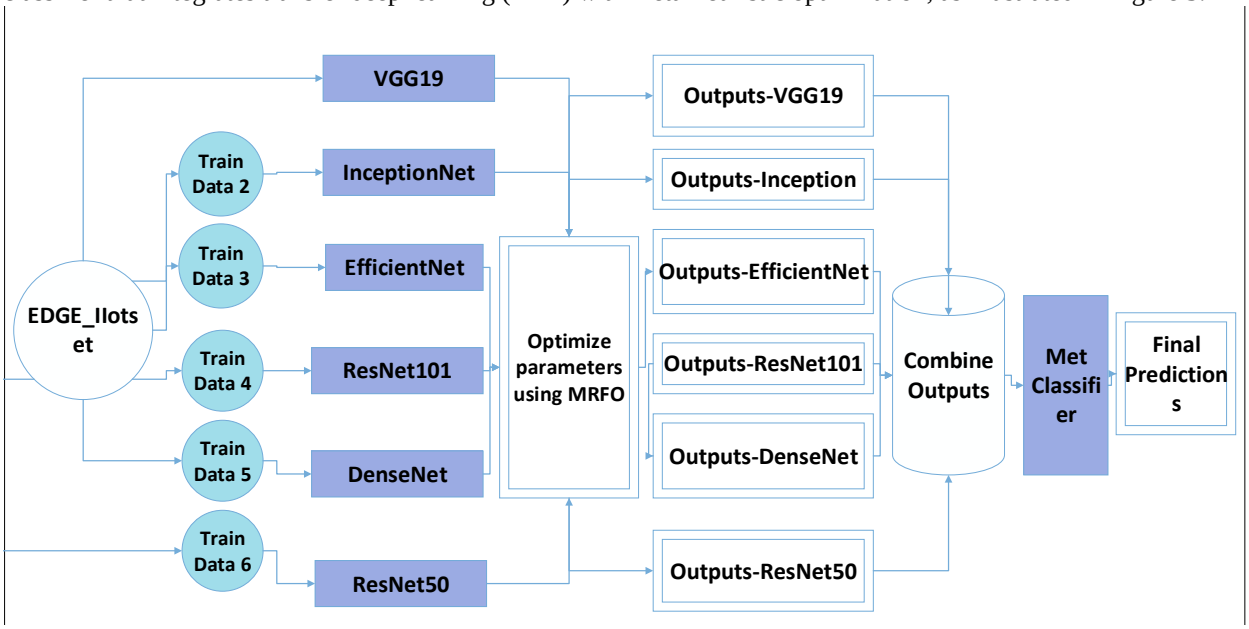

Fig.3. The Proposed MRFO-MoENet framework

The central idea of MRFO-MoENet is to construct a multi-layer classification system that leverages pretrained deep models as expert learners and refines their integration using the Manta Ray Foraging Optimization (MRFO) algorithm. The model is tailored to detect and classify various types of intrusions in IoT environments, using the Edge-IIoTset dataset as a benchmark.

The motivation for using transfer learning is to capitalize on pre-existing knowledge acquired from large-scale image datasets and adapt it to the IoT intrusion domain, thereby improving learning efficiency and reducing training costs. Furthermore, the ensemble strategy helps to enhance model robustness and classification accuracy by fusing diverse feature representations extracted from multiple expert networks.
In the proposed approach:

- First, the raw data from the Edge-IIoTset dataset undergoes preprocessing and feature engineering.
- Next, the processed data is input into a series of pretrained CNN-based models (experts), whose outputs are fused to create a richer feature set.
- These outputs, representing **probability vectors across 14 classes**, are then optimized via the **MRFO algorithm**, which tunes the hyperparameters to maximize classification performance.
- Finally, the optimized outputs are passed through a high-level meta-classifier (e.g., Random Forest or Gradient Boosting) to generate the final predictions.

As shown in **Figure 3**, the entire architecture consists of the following three main layers:

1. Expert Model Construction: Extraction of discriminative features using pretrained CNN models on the Edge-IIoTset dataset.
2. Optimization Layer: Fine-tuning of hyperparameters and feature selection using the MRFO algorithm to enhance classification performance.
3. Meta-Classification Layer: Application of a high-level ensemble classifier to produce the final intrusion detection outcome.

3.2.1 Construction of Expert Models Using Transfer Learning

In the proposed MRFO-MoENet framework, a diverse set of pretrained Convolutional Neural Networks (CNNs) are utilized as expert models. These networks serve as the foundational learners in a Mixture of Experts (MoE) architecture, each contributing unique and complementary feature representations to improve the generalization capability and robustness of the intrusion detection system.

DOI: 10.46121/pspc.54.1.24

Specifically, **six deep transfer learning models** are fine-tuned on the Edge-IIoTset dataset:

- VGG19: A classic and powerful CNN architecture trained on ImageNet, comprising 19 weight layers, known for its simplicity and depth, ideal for feature extraction.
- EfficientNetV7: An advanced architecture by Google optimized for both speed and accuracy using compound scaling.
- ResNet101 and ResNet50: Deep residual networks that leverage skip connections to enable the training of very deep networks without gradient vanishing.
- DenseNet121: A densely connected CNN where each layer receives input from all previous layers, enhancing feature reuse and gradient flow.
- InceptionV3: A model that utilizes multi-scale convolution filters within the same layer, making it highly effective in capturing different spatial hierarchies in the input data.

Each expert model is initialized with pretrained weights from ImageNet and fine-tuned on the IoT-specific dataset to adapt their parameters to the characteristics of Edge-IIoTset data. Feature vectors extracted from the penultimate layers of these networks are used as input for the optimization and fusion stages.

The following pseudocode illustrates the training and ensemble process for the expert models:

Algorithm 1: Construction and Optimization of Expert Models

Input	Preprocessed Edge-IIoTset Dataset
	function Train_Expert_Models(dataset):
1	*# Step 1: Initialize pretrained models*
2	*model_1 = VGG19(dataset)*
3	*model_2 = EfficientNetV7(dataset)*
4	*model_3 = ResNet101(dataset)*
5	*model_4 = ResNet50(dataset)*
6	*model_5 = DenseNet121(dataset)*
7	*model_6 = InceptionV3(dataset)*
8	
9	*# Step 2: Fine-tune each model on Edge-IIoTset*
10	*base_models = [model_1, model_2, model_3, model_4, model_5, model_6]*
11	*optimized_models = []*
12	
13	*for model in base_models:*
14	*# Step 3: Optimize model hyperparameters using MRFO*
15	*optimized_params = MRFO(objective_function=model.validation_accuracy)*
16	*model.set_params(optimized_params)*
17	*optimized_models.append(model)*
18	
19	*# Step 4: Predict using each optimized model*
20	*predictions = []*
21	*for model in optimized_models:*
22	*pred = model.predict(dataset.test)*
23	*predictions.append(pred)*
24	*# Step 5: Combine predictions from all expert models*
25	*final_output = Aggregate_Predictions(predictions)*
26	*return final_output*
Output	*Optimized predictions from expert models*

Fig. 4. Algorithm 1: Construction and Optimization of Expert Models

DOI: 10.46121/pspc.54.1.24

3.2.2 Optimization of Hyperparameters Using MRFO Algorithm

In this section, the **Manta Ray Foraging Optimization (MRFO)** algorithm is used to fine-tune the hyperparameters of the expert models in the MRFO-MoENet framework. Optimal hyperparameter selection is crucial for achieving high performance in deep learning models, especially in ensemble structures with multiple transfer learning components. Traditional hyperparameter tuning approaches often rely on manual search or grid search, which are computationally expensive, inflexible, and impractical for complex deep architectures like the one used in this study. Moreover, manual methods may fail to generalize well across different datasets. Therefore, a robust and adaptive optimization strategy like MRFO is employed. Important Hyperparameters Considered:

1. Learning Rate: Controls the speed at which the model updates weights. Improper settings can either lead to poor convergence (too high) or unnecessarily slow training (too low).
2. Dropout Rate: Regularizes the network by randomly disabling a fraction of neurons, helping prevent overfitting.
3. Number of Dense Layers: Affects the model's ability to learn hierarchical representations.
4. Number of Neurons per Dense Layer: Directly influences the network capacity and computational complexity.
5. Batch Size: Determines the number of samples used for each gradient update; influences memory usage and convergence.
6. Number of Epochs: The number of full passes over the dataset; critical for model convergence.
7. Activation Functions and Optimizer Type (in some configurations): Additional tunable elements for network dynamics.

Each expert model (e.g., VGG19, ResNet101, DenseNet121, etc.) requires an individualized set of hyperparameters. MRFO is applied to each model independently to determine the best settings that maximize classification performance.

3.2.2.1 MRFO-Based Optimization Strategy

MRFO is a nature-inspired metaheuristic algorithm that mimics the intelligent foraging behavior of manta rays. The optimization process is modeled in three main phases:

1. Chain Foraging: Manta rays follow the best individuals and move collectively while exploring the search space. In this stage, each candidate solution updates its position based on the best solution so far.

2. Cyclone Foraging: This phase mimics the spiral swimming behavior of manta rays to explore diverse regions of the search space, thus avoiding premature convergence.

3. Somersault Foraging: Manta rays occasionally perform somersault-like movements to jump around the search space and escape local optima.

The general structure of MRFO for hyperparameter tuning follows a population-based optimization process that iteratively updates candidate solutions to find the optimal hyperparameters.

- Candidate Solution: Each manta ray represents a potential set of hyperparameters for a given expert model.
- Fitness Function: Classification accuracy (or validation loss) of the deep learning model on a hold-out validation set.
- Search Space: Defined by permissible ranges of hyperparameters (e.g., learning rate ∈ [1e-5, 1e-1], dropout ∈ [0.1, 0.5], epochs ∈ [10, 100], etc.)

The algorithm steps define a sequential process through which the MRFO (Manta Ray Foraging Optimization) algorithm performs hyperparameter tuning by initializing a population, evaluating fitness, and updating solutions using foraging strategies until convergence.

1. Initialize the Population: Randomly generate a set of candidate solutions (manta rays), each encoding a set of hyperparameters.
2. Evaluate Fitness: Train the expert model using each solution and compute its accuracy on the validation set.
3. Chain Foraging Phase: Update each solution based on the best-performing candidate.
4. Cyclone Foraging Phase: Enhance diversity by moving each solution in a spiral path toward the global best.
5. Somersault Foraging Phase: Reposition candidates around the best-known solution to avoid local minima.
6. Update Global Best: Track the best hyperparameter combination found so far.

DOI: 10.46121/pspc.54.1.24

7. Termination: Repeat the above steps until a stopping criterion (e.g., maximum iterations or convergence) is met.

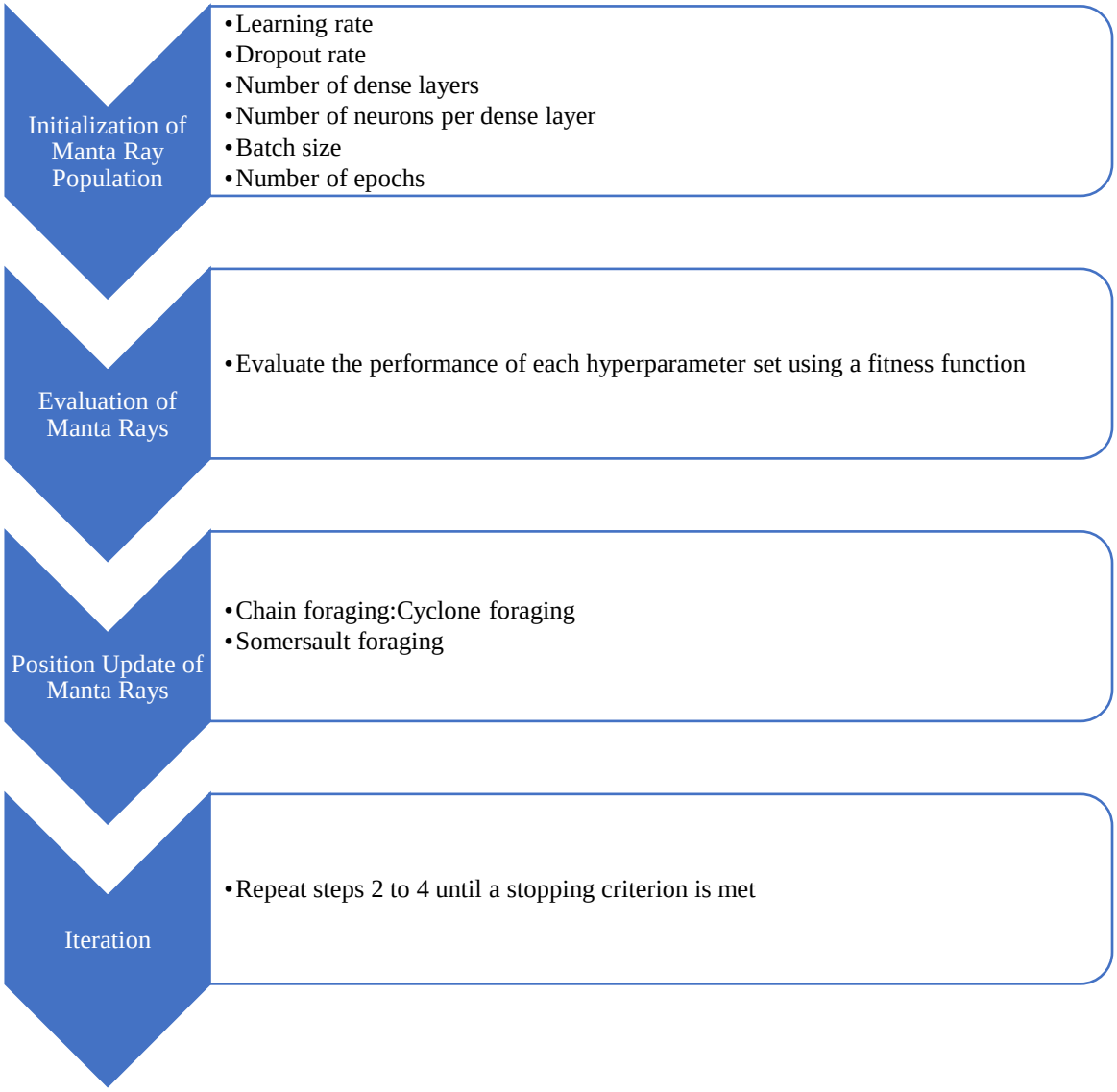

Fig. 5. Algorithm steps of MRFO

To effectively evaluate and guide the optimization process of hyperparameters in our MRFO-MoENet model, a custom fitness function is defined. This function integrates multiple key performance metrics into a single scalar value to ensure balanced learning performance across all critical aspects of classification. The proposed fitness function is formulated as follows:

(1)

Where:

- F1: The F1-score of the model, representing the harmonic mean of precision and recall. Using (1−F1) encourages maximization of the F1-score.
- FPFPFP: The number of false positives—normal instances incorrectly classified as attacks.
- FN: The number of false negatives—attacks incorrectly classified as normal traffic.
- N: Total number of samples in the dataset.
- : The actual training time required by the model.
- : The maximum allowed or observed training time.

DOI: 10.46121/pspc.54.1.24

- α, β, γ: Weight coefficients that determine the relative importance of classification performance, error rate, and training efficiency, respectively.

This multi-objective fitness function ensures that:

- The model achieves high predictive performance (via F1).
- Both false positives and false negatives are minimized, which is crucial in intrusion detection systems.
- The computational efficiency is considered by penalizing models that require excessive training time.

By integrating these three critical dimensions—accuracy, reliability, and efficiency—the fitness function drives the metaheuristic optimization algorithm (MRFO) toward finding a balanced and optimal configuration of hyperparameters.

Algorithm 2: Manta Ray Foraging Optimization (MRFO) for Hyperparameter Tuning

Input	N: Population size (number of candidate solutions) T: Maximum number of iterations D: Dimensionality of the hyperparameter search space
1	Initialize a population with random solutions in the hyperparameter space
2	Evaluate the fitness of each solution using the defined fitness function
3	Set the best solution ▷ Represents the best hyperparameter combination
4	For t = 1 to T do
5	For t = 1 to T do
6	For each solution Xi in the population do
7	Generate random value r ∈ [0, 1]
8	
9	If r < 1/3 then ▷ Chain foraging phase
10	Update Xi using:
11	
12	
13	Else if 1/3 ≤ r < 2/3 then ▷ Cyclone foraging phase
14	Compute spiral factor
15	Update using:
16	
17	
18	Else ▷ Somersault foraging phase
19	Generate somersault factor S ∈ [0, 2]
20	Update Xi using:
21	
22	
23	End For
24	
25	Evaluate updated solutions and update if a better solution is found
26	End For
27	
28	Return: Best hyperparameter configuration and its corresponding fitness value
Output	Optimal hyperparameter set (best solution) Minimum fitness value achieved through MRFO iterations

Fig. 6. Pseudocode of the Manta Ray Foraging Optimization (MRFO) Algorithm

3.2.3 Final Classification Using the Meta-Classifier

In the final stage of the proposed MRFO-MoENet framework, a high-level meta-classifier is employed to produce the final classification output based on the features extracted and optimized by the ensemble of deep transfer learning models. For this purpose, the **Random Forest (RF)** algorithm is utilized due to its strong generalization ability, resistance to overfitting, and robustness to noisy data and high-dimensional feature spaces.

DOI: 10.46121/pspc.54.1.24

The RF model operates as a collection of decision trees, each trained on a bootstrap sample of the input data and a randomly selected subset of features. The final decision is made by aggregating the predictions of all trees through majority voting, which contributes to reducing variance and improving prediction accuracy.
The key steps of using Random Forest as a meta-classifier involve training multiple base classifiers, collecting their predictions, and then using these predictions as input features to train the Random Forest, which learns to combine the base outputs for final decision-making.

1. Input Construction: The final input to the RF classifier is the concatenated output vectors from the optimized base classifiers (e.g., Softmax probability scores from each transfer learning model).
2. Tree Ensemble Creation: Multiple decision trees are built independently using random subsets of features and samples drawn via bootstrapping.
3. Final Decision Making: The output class for each input instance is determined by majority voting across the ensemble of decision trees.

Table 1: Hyperparameters of random forest

Hyperparameter	Description
n_estimators	The number of trees in the ensemble. A higher value improves stability and accuracy but increases computation cost.
max_features	The number of features to consider when looking for the best split at each node.
max_depth	Maximum allowed depth of the trees. A smaller depth reduces overfitting but may lead to underfitting.
min_samples_split	Minimum number of samples required to split an internal node.
min_samples_leaf	Minimum number of samples required to be at a leaf node.
bootstrap	Whether bootstrap samples are used when building trees. Set to **True** to enable sampling with replacement.
criterion	The function used to evaluate the quality of a split. Common options are **"gini"** for Gini impurity and **"entropy"** for information gain.

This meta-classification layer serves as the final decision-maker that integrates the optimized knowledge extracted from all expert DTL models. The combination of feature diversity and hyperparameter tuning significantly improves the classification performance on the IoT intrusion detection task, particularly in multiclass scenarios like the Edge-IIoTset dataset.

EXPERIMENTS AND RESULTS

This section presents the experimental setup and results for evaluating the proposed MRFO-MoENet model. It begins with a description of the dataset (Section 4.1), followed by the preprocessing steps applied to the data (Section 4.2). Section 4.3 outlines the evaluation metrics used to assess the model's performance. In Section 4.4, the MRFO algorithm is applied to optimize the hyperparameters of the base classifiers. Section 4.5 provides the performance results of the proposed method. Finally, Section 4.6 compares the proposed model against several baseline and state-of-the-art methods. The entire experimental flow, from input data to final classification and comparison, is depicted in Figure 7.

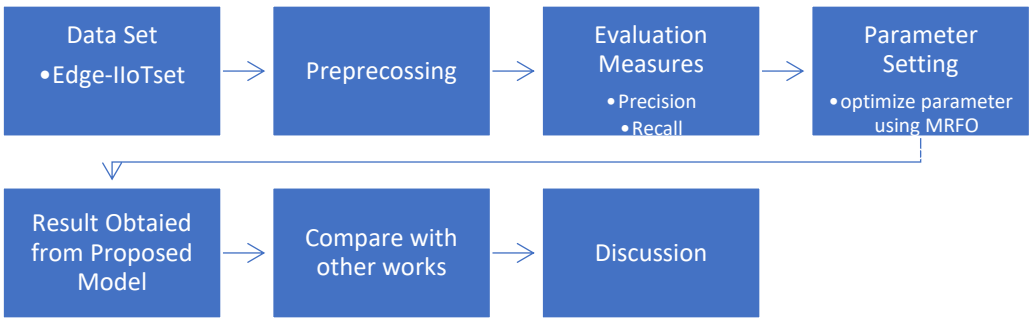

Fig. 7. Evaluation Process of proposed method

DOI: 10.46121/pspc.54.1.24

Power System Protection and Control
ISSN:1674-3415

4.1 Data set

In the proposed framework, we have used the cyber security dataset named Edge-IIoTset to implement and evaluate the proposed method. This dataset was produced by developing a real IIoT environment containing more than 10 types of Internet of Things devices. This dataset contains 14 attack categories related to IIoT connection protocols. Figures 8 shows the number of samples for each class.

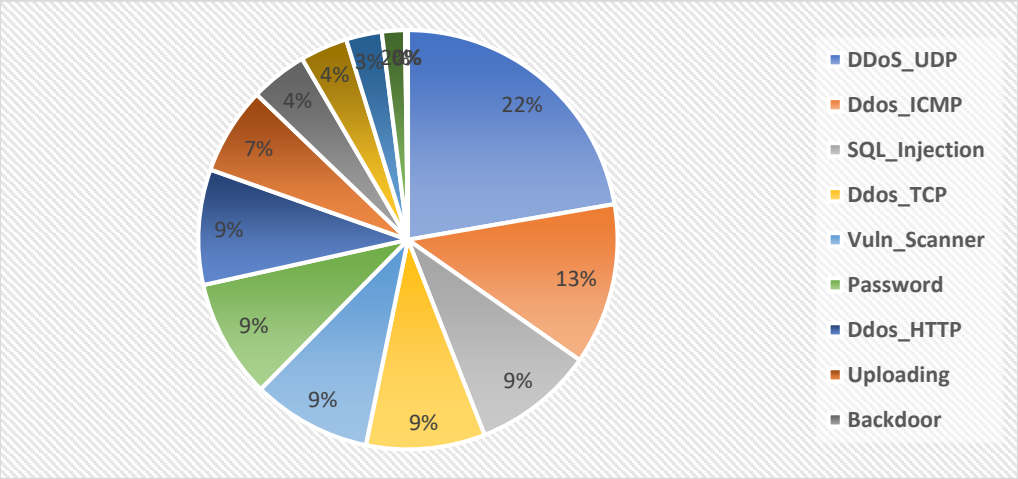

Fig. 8. Class Distribution in the Original Edge-IIoTset Dataset

As it is clear in figure 8, it shows the frequency percentage of each sample in the classes, which indicates the imbalance of the data set and this problem should be solved. This problem will be solved in the next section.

4-2 Preprocessing

One of the primary challenges encountered in this dataset pertains to the class imbalance problem. Specifically, a significant disparity exists in the number of samples among different classes. For instance, the DDoS_UDP class contains 121,567 instances—accounting for approximately 22% of the entire dataset—whereas minority classes such as MITM and Fingerprinting comprise only 358 and 853 samples, respectively, representing less than 1% of the total. This imbalance adversely affects the overall classification accuracy by biasing the model toward majority classes. To address this issue, data balancing was performed through a meta-sampling approach. In particular, oversampling was employed to generate synthetic instances for underrepresented classes. To this end, the Generative Adversarial Network (GAN) oversampling technique was utilized, which not only balances the dataset but also mitigates the risk of sample redundancy. Figure 8 presents the class distribution in the original dataset, while Figure 9 shows the distribution after applying the balancing strategy.

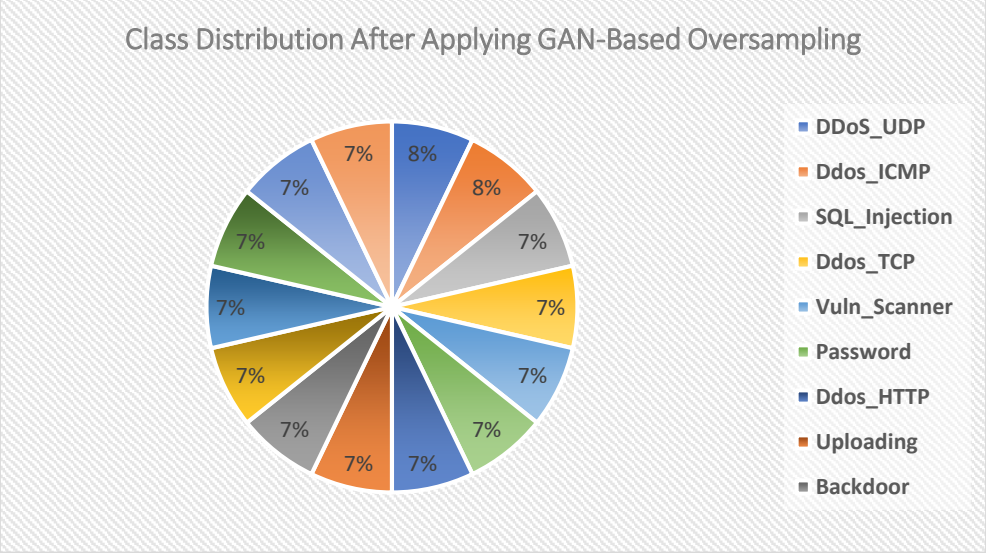

Fig. 9. Class Distribution After Applying GAN-Based Oversampling

DOI: 10.46121/pspc.54.1.24

Figure 9 illustrates that post-balancing, the total number of data set samples increased to 8683, indicating that there are 121,567 data points for each class.

4-3 Evaluation Measures

This section introduces three criteria: detection accuracy, precision, and recall. We have employed the following three criteria for assessment, as delineated below (Equation (3) to (6)):

(3)

(4)

(5)

(6)

FP and FN denote the wrong sample of each class, while TP and TN denote the correct diagnoses for each class. Undoubtedly, TP and TN values closer to one another indicate a better model. The confusion matrix's main diameter contains the TP and TN values.

4-4 Setting up Hyperparameters

4.4.1 Setting the Transfer Deep Learning Parameters Using MRFO Algorithm

In the proposed methodology, the optimization of critical hyperparameters such as learning rate, dropout rate, number of dense layers, and other architectural settings has a significant impact on the classification performance of the transfer learning models. To achieve optimal performance, the Manta Ray Foraging Optimization (MRFO) algorithm has been utilized for tuning these parameters. MRFO mimics the foraging strategies of manta rays and effectively balances exploration and exploitation in the search space.

Each hyperparameter is treated as a decision variable in the MRFO search process. In each iteration, the MRFO algorithm generates new candidate solutions (parameter sets) which are then used to configure and train transfer learning models. The validation accuracy obtained from each configuration is evaluated as the fitness value, guiding the optimization towards the most effective parameter combination.

The MRFO algorithm was implemented in Python, and for each of the deep transfer learning networks, including VGG19, EfficientNetV7, ResNet101, ResNet50, DenseNet121, and InceptionV3, the optimal values of the hyperparameters were determined. The table 2 presents the optimal values obtained through the MRFO algorithm:

Table 2. **Optimal Hyperparameters Extracted Using the Proposed MRFO Algorithm**

Network	Learning Rate	Dropout Rate	# Dense Layers	Neurons per Dense Layer	Batch Size	Epochs
VGG19	0.0015	0.15	2	256	128	100
EfficientNetV7	0.001	0.18	3	256	128	100
ResNet101	0.001	0.15	2	128	64	100
ResNet50	0.002	0.15	2	256	128	120
DenseNet121	0.002	0.14	3	256	128	100
InceptionV3	0.001	0.12	2	256	128	100

As demonstrated in the table, the MRFO algorithm effectively explores the defined search ranges: learning rate in [0.001–0.003], dropout rate in [0.1–0.5], dense layers in [1–3], neurons per layer in [64–512], and batch size in [16–128]. This adaptive tuning ensures that each model is fine-tuned for optimal performance, leveraging the global search capabilities of MRFO to find superior configurations.

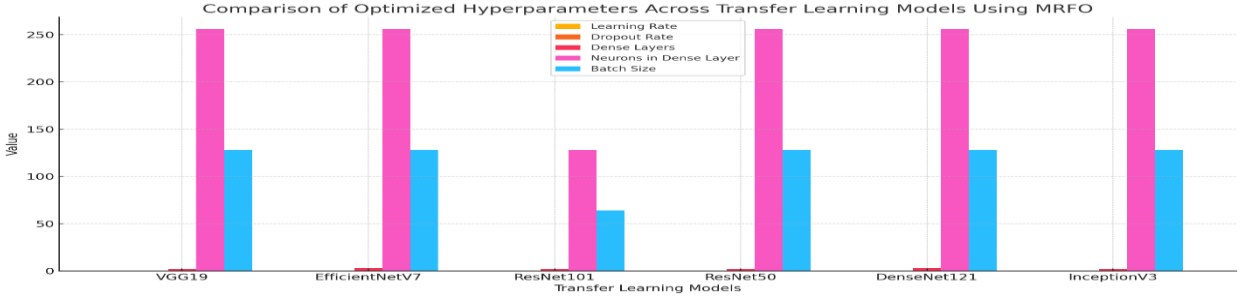

Fig.10. Comparison of Optimized Hyperparameters Across Transfer Learning Models Using MRFO

DOI: 10.46121/pspc.54.1.24

Aiming at finding the optimal combination of hyperparameters for different transfer learning models, the MRFO algorithm was able to effectively determine the appropriate values of variables such as learning rate, dropout rate, number of dense layers, number of neurons per layer, batch size, and number of epochs for each model. Analysis of the obtained optimal values shows that:

• Learning Rate: The optimal learning rate value for most models is in the range of 0.001 to 0.002. This relatively small value allows the models to move with stable and controlled steps during weight updates and avoids severe fluctuations. Both the EfficientNetV7 and ResNet101 models used a lower learning rate (0.001), indicating a higher sensitivity of these architectures to fine-tuning of the learning rate.

• Dropout rate: The dropout rate in most models is set between 0.12 and 0.18, which represents an attempt to strike a balance between reducing overfitting and maintaining the learning ability of the model. The DenseNet121 model performed optimally with a dropout of 0.14, which is consistent with the more complex nature of this network.

• Number of Dense layers and neurons: Architectures such as EfficientNet and InceptionV3 used 3 Dense layers with 256 neurons per layer. This configuration allowed for richer feature extraction. In contrast, ResNet101 only required 2 Dense layers with 128 neurons, which reflects the inherent power of this architecture in learning more abstract features.

• Batch size: Most models used a Batch size of 128, except for ResNet101, which had a value of 64. The use of a smaller batch in this model may be due to limited memory or its deeper structure that adapts better to smaller values.

• Number of Epochs: The number of epochs in most models is set to 100. Only the ResNet50 model used 120 epochs, indicating that this network needs more training sessions for proper convergence.

4-4-2 Setting the random forest parameters

As part of the final classification stage in the proposed methodology, the hyperparameters of the Random Forest classifier were optimized to maximize performance. To achieve this, several key parameters were investigated, including the number of decision trees (n_estimators), the splitting criterion (criterion), the maximum depth of trees (max_depth), and the number of features considered at each split (max_features). To find the optimal configuration, the MRFO (Manta Ray Foraging Optimization) algorithm was employed as the search mechanism. A series of experiments were conducted in which different values for these parameters were tested simultaneously. The splitting criteria Gini and Entropy were both evaluated, and the maximum depth of the trees was varied between 2 and 10. The fitness of each configuration was measured based on classification accuracy. Table 3 summarizes the optimal hyperparameters selected for the Random Forest classifier using the MRFO algorithm. In addition, Figure 11 presents a comparative visualization of accuracy results across various parameter configurations. The chart clearly demonstrates the advantage of utilizing the optimized settings derived from MRFO in enhancing the final prediction performance.

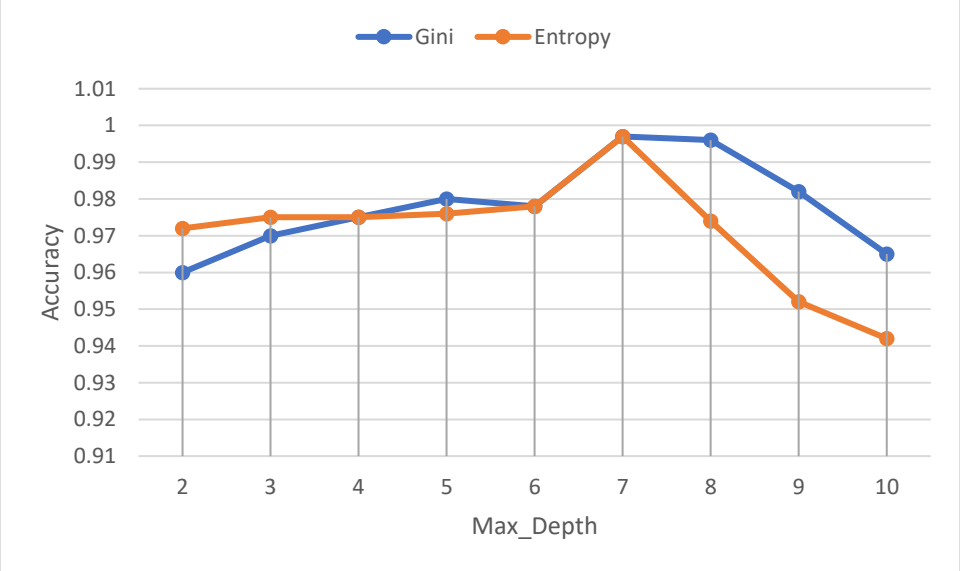

Fig. 11. The effect of Gini and entropy parameters with different Max_Depths on the data set

As illustrated in Figure 11, there is a positive correlation between the max_depth parameter and the classification accuracy up to a certain point. Specifically, the accuracy improves as the depth increases, peaking when

DOI: 10.46121/pspc.54.1.24

max_depth = 7 using the Gini criterion. Beyond this point, a decline in accuracy is observed, likely due to overfitting caused by excessive tree depth. Therefore, 7 is identified as the optimal value for the max_depth parameter.

The remaining hyperparameters were also empirically evaluated through multiple experimental iterations, and the optimal values were determined accordingly. Table 3 summarizes the best-performing configuration for the Random Forest classifier obtained using the MRFO algorithm.

Table 3: Hyperparameter values obtained for random forest

Hyper Parameter	Value
n_estimators	**100**
max_features	**Log2**
max_depth	**7**
min_samples_split	**2**
min_samples_leaf	**1**
bootstrap	**True**
criterion	**Gini**

4-4-3 The results obtained from the experiments

This section presents the outcomes derived from the experimental phase of the proposed method. To ensure robust evaluation, a 5-fold cross-validation approach was employed during testing. The performance of the model on the experimental dataset is illustrated through the confusion matrix in Figure 12, which demonstrates the classification performance across various classes. Additionally, Table 4 summarizes the accuracy scores obtained on both the training and testing datasets. To provide a more comprehensive assessment, three critical performance metrics—Precision, Recall, and F1-Score—have been analyzed. The corresponding results are visualized in Figures 13, 14, and 15, respectively. These metrics highlight the classification effectiveness of the proposed hybrid architecture, offering insights into its ability to correctly identify both majority and minority class instances in an imbalanced dataset scenario.

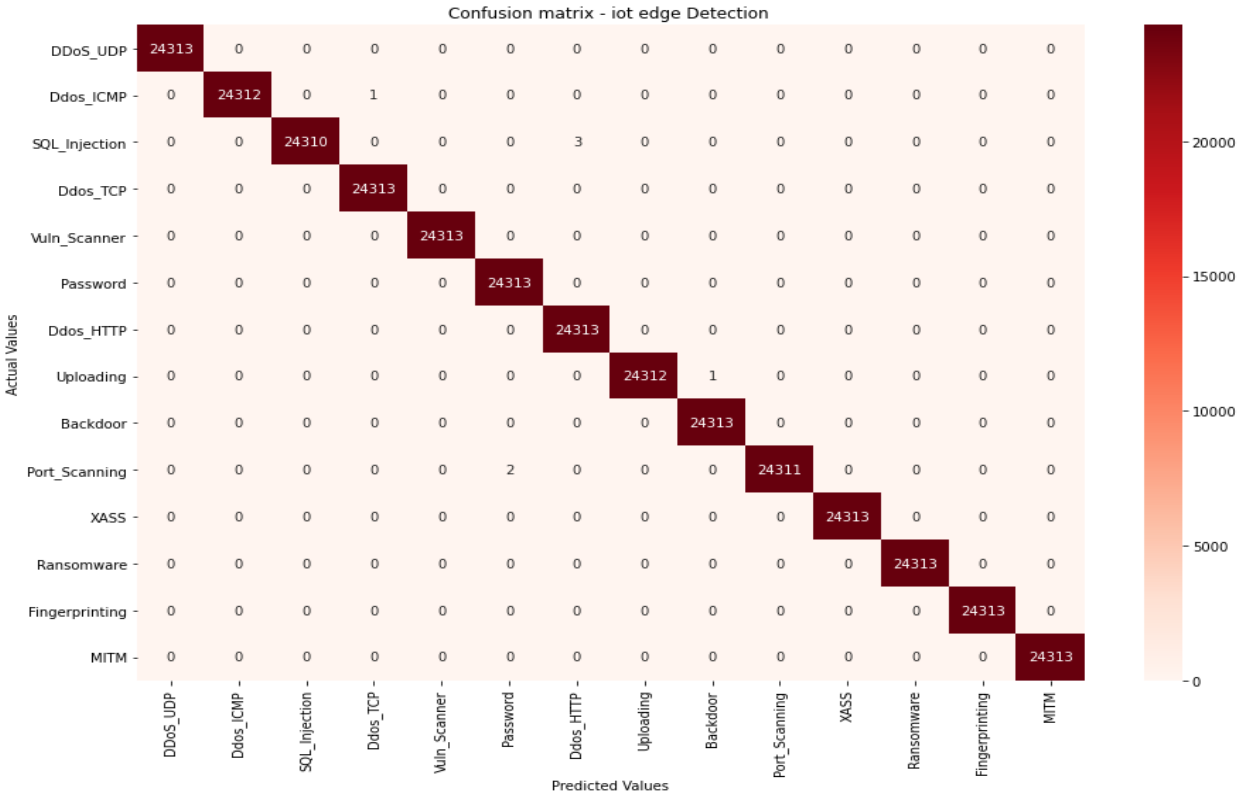

Fig.12. The confusion matrix on the final model for the data set

DOI: 10.46121/pspc.54.1.24

Power System Protection and Control
ISSN:1674-3415

Table 4: The final accuracy obtained on the training and testing dataset

	Test Set(%)	Training Set(%)
Proposed Model	99.92	100

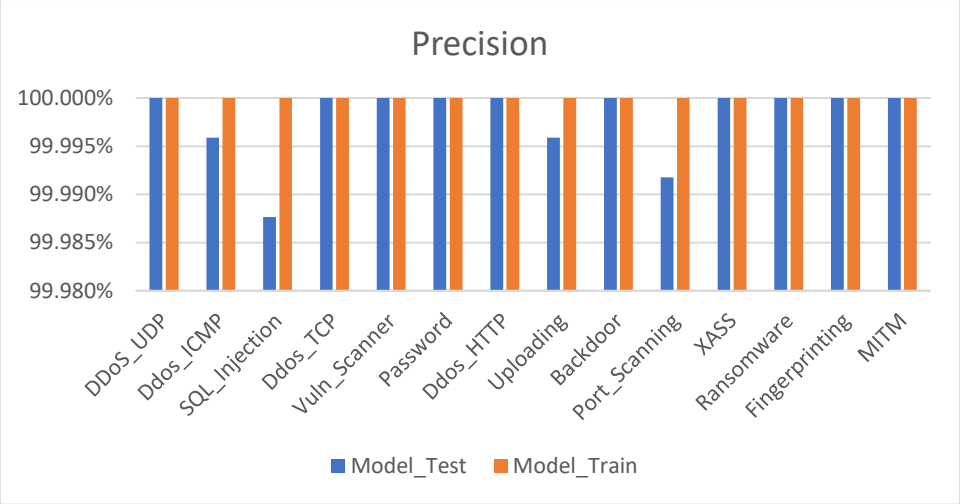

Fig.13. Precision value obtained on the data set for training and testing data

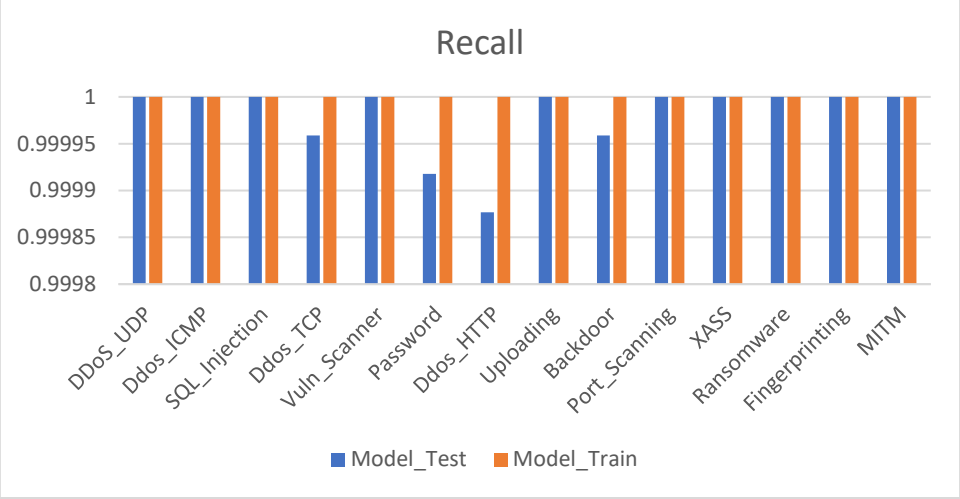

Fig. 14. Recall value obtained on the data set for training and testing data

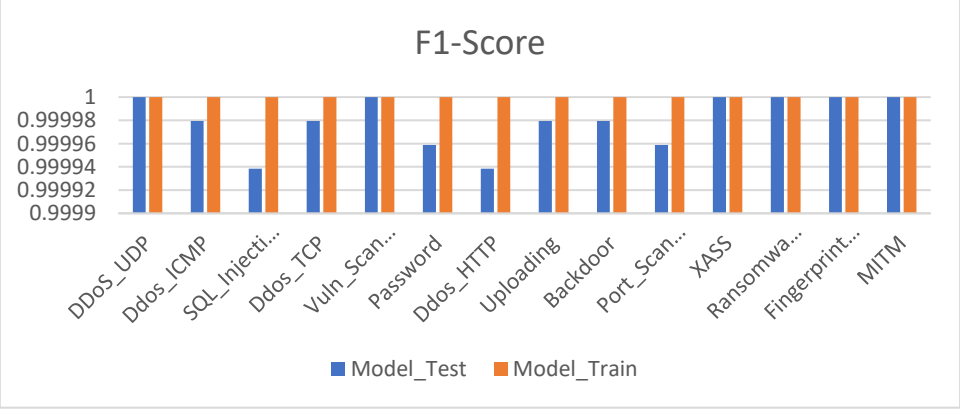

Fig. 15. F1-Score value obtained on the data set for training and testing data

DOI: 10.46121/pspc.54.1.24

Power System Protection and Control
ISSN:1674-3415

As illustrated in Figures 13, 14, and 15, the performance metrics—accuracy, recall, and F1-score—have been reported for both the training and testing datasets. In the training dataset, all three metrics achieved a perfect score of 100% across all classes, indicating excellent learning capability and generalization during model training. In contrast, the testing dataset also shows an overall accuracy of 100%, with a slight deviation (~99.99%) observed in the classes "Ddos_ICMP", "SQL_Injection", "Uploading", and "Port_Scanning". Regarding the recall metric, all classes again achieved 100%, except for "Ddos_TCP", "DDos_HTTP", "Password", and "Backdoor", where the recall was approximately 99.99%. Similarly, the F1-score was 100% for most classes in the experimental dataset, with a marginal drop (~99.99%) noted for the same classes mentioned above: "Ddos_TCP", "DDos_HTTP", "Password", "Backdoor", "Ddos_ICMP", "SQL_Injection", "Uploading", and "Port_Scanning". These outstanding results demonstrate the effectiveness of the proposed approach, particularly the use of optimized hyperparameters derived from the metaheuristic group-based classification model. By tuning key parameters, the model significantly reduces both false positives (FP) and false negatives (FN), thereby enhancing overall prediction accuracy. Furthermore, Figures 16 and 17 depict the trends of accuracy and loss over training epochs. Two strategies—"min_loss" and "max_acc"—were evaluated to identify the most effective configuration. The model achieved its optimal state at epoch 25, where the lowest loss was recorded, justifying its selection as the final trained model.

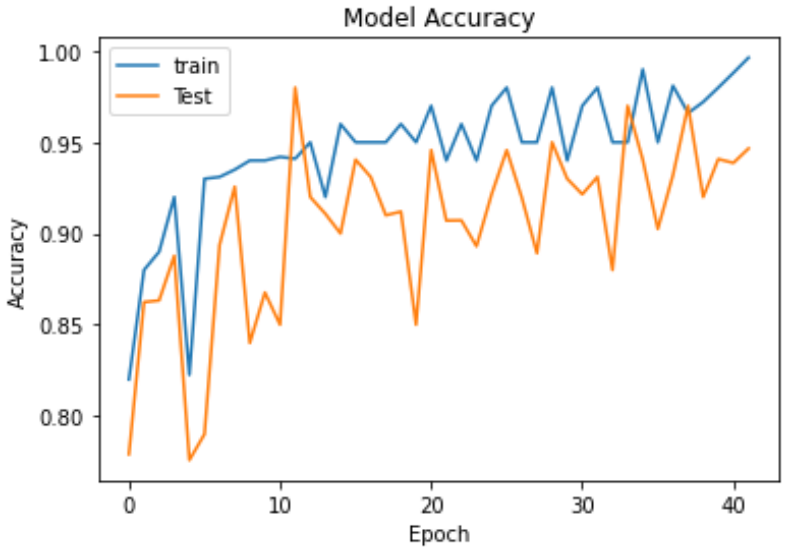

Fig. 16. The accuracy measure in different epochs in Ensemble optimized DTL model

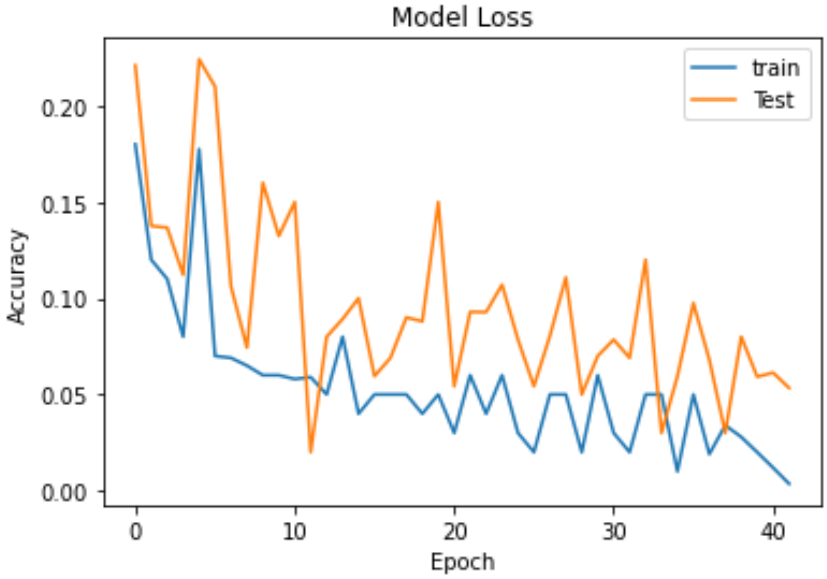

Fig. 17. The Loss measure in different epochs in Ensemble optimized DTL model

DOI: 10.46121/pspc.54.1.24

4-4-4 Comparison with Other Works

In recent years, numerous methods have been proposed to enhance intrusion detection in IoT environments, leveraging a variety of machine learning (ML), deep learning (DL), and hybrid approaches. Despite these advancements, many of the existing methods still face challenges such as limited generalizability, low accuracy on large-scale or heterogeneous datasets, or insufficient optimization strategies. To evaluate the effectiveness of the proposed method, a comparative analysis has been conducted against several recent state-of-the-art techniques. **Table 5** presents this comparison, detailing the dataset used, method applied, and the achieved accuracy in each case.

Table 5. Comparison of the proposed method with state-of-the-art approaches

Ref (Year)	Dataset	Method	Accuracy
[18], 2024	ROUT-4-2023	Hybrid Intrusion Detection System for RPL IoT Networks Using ML and DL	95.0%
[19], 2023	CICIDS2017	Deep Learning Model for IoT Intrusion Detection Systems	95.0%
[20], 2024	UNSW-NB15, CIC-IDS2018, CIC-IOT2023	Deep Learning-Based Network Intrusion Detection for IoT	98.2%
[21], 2024	BoT-IoT, CSE-CIC-IDS2018	End-to-End Intrusion Detection System Using Deep Learning	99.2%
[22], 2021	KDDCup-99, NSL-KDD, BoT-IoT, CICIDS2017	IoT IDS Using Deep Learning and Enhanced Optimization	96.7%
[23], 2020	BoT-IoT	IoT Intrusion Detection with Deep Learning	95.4%
[24], 2022	BoT-IoT	Deep Learning for IoT Intrusion Detection	96.7%
Proposed	**Edge-IIoTset**	**Hybrid Ensemble Classifier Optimized Using MRFO Algorithm**	**99.91%**

As observed, the proposed method significantly outperforms previous techniques in terms of accuracy, achieving **99.91%** on the **Edge-IIoTset** dataset—a challenging and modern benchmark for IoT security evaluation. This high accuracy is attributed to the synergy between **transfer learning**, **ensemble classification**, and parameter optimization via the **Marine Predators Feeding Optimization (MRFO)** algorithm. The MRFO algorithm efficiently tunes critical hyperparameters, leading to reduced error rates and improved generalization.

By contrast:

- **Method [18]** achieves 95% on ROUT-4-2023, which, although useful for detecting routing-specific anomalies in IoT networks, does not generalize well across diverse threat landscapes.
- **Method [19]**, applied to the CICIDS2017 dataset, also reports 95% accuracy, but lacks robustness against modern, heterogeneous IoT attack types.
- **Method [20]** shows improved results (98.2%) across multiple datasets; however, the absence of a unified optimization mechanism limits its performance consistency.
- **Method [21]** reports high accuracy (~99.2%), but does not explicitly address hyperparameter tuning or adaptability across unseen datasets.
- **Older approaches** such as [22], [23], and [24] represent important milestones in the field but are constrained by either outdated datasets or non-optimized learning pipelines.

In conclusion, the **proposed hybrid ensemble model with MRFO-based optimization** demonstrates superior accuracy and reliability, confirming its potential as a robust and scalable solution for intrusion detection in next-generation IoT networks. Its strong performance on the **Edge-IIoTset** dataset also highlights its capability to address contemporary cybersecurity challenges effectively.

CONCLUSIONS AND FUTURE WORKS

In this paper, a novel method was proposed for intrusion detection in IoT environments by combining deep transfer learning with an optimized ensemble classification approach. The Marine Predators Feeding Optimization (MRFO) algorithm was employed to fine-tune key hyperparameters of deep neural networks, leading to improved classification accuracy and reduced false alarms. By leveraging pre-trained models, the system achieved high performance with less training data. Experimental results on the Edge-IIoTset dataset demonstrated the effectiveness of the proposed method, reaching an accuracy of 99.91%. This indicates the model's potential for

DOI: 10.46121/pspc.54.1.24

real-world applications in IoT security, especially in handling various types of cyberattacks with high precision and reliability.

For future research, expanding and diversifying datasets is essential to improve model generalizability. Exploring new or hybrid optimization algorithms could further enhance learning efficiency. Additionally, integrating the model with other security mechanisms, such as anomaly detection, and focusing on real-time implementation for edge devices are important next steps. Finally, testing the model in diverse network environments and on different IoT devices will help assess its adaptability and robustness. In summary, the proposed framework offers a promising direction for secure and intelligent IoT systems, and with further development, it can play a critical role in strengthening future IoT infrastructures.

REFERENCES

1. Y. Yang, L. Wu, G. Yin, L. Li and H. Zhao, "A survey on security and privacy issues in Internet of Things," IEEE Internet of things Journal, vol. 4, no. 5, pp. 1250--1258, 2017.
2. A. Heidari and M. A. Jabraeil Jamali, "Internet of Things intrusion detection systems: a comprehensive review and future directions," Cluster Computing, vol. 26, no. 6, pp. 3753--3780, 2023.
3. A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges," Archives of Computational Methods in Engineering, vol. 28, no. 4, pp. 3211--3243, 2021.
4. R. Ahmad and I. Alsmadi, "Machine learning approaches to IoT security: A systematic literature review," Internet of Things, vol. 14, p. 100365, 2021.
5. S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed and R. Islam, "Dependable intrusion detection system for IoT: A deep transfer learning based approach," IEEE Transactions on Industrial Informatics, vol. 19, no. 1, pp. 1006--1017, 2022.
6. A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," Future Generation Computer Systems, vol. 82, pp. 761--768, 2018.
7. F. S. Gharehchopogh, B. Abdollahzadeh, S. Barshandeh and B. Arasteh, "A multi-objective mutation-based dynamic Harris Hawks optimization for botnet detection in IoT," Internet of Things, vol. 24, p. 100952, 2023.
8. B. Kaur, S. Dadkhah, F. Shoeleh, E. C. P. Neto, P. a. I. S. Xiong, P. Lamontagne, S. Ray and A. A. Ghorbani, "Internet of things (IoT) security dataset evolution: Challenges and future directions," Internet of Things, vol. 22, p. 100780, 2023.
9. I. H. Sarker, A. I. Khan, Y. B. Abushark and F. Alsolami, "Internet of things (iot) security intelligence: a comprehensive overview, machine learning solutions and research directions," Mobile Networks and Applications, vol. 28, no. 1, pp. 296--312, 2023.
10. X. Chen, R. Yang, Y. Xue, M. Huang, R. Ferrero and Z. Wang, "Deep transfer learning for bearing fault diagnosis: A systematic review since 2016," IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1--21, 2023.
11. P. a. A. A. a. L. P.-P. a. R. M. a. S. G. A. a. G. B. F. a. S. T. Yan, "A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: Methods, applications, and directions," IEEE Access, 2024.

DOI: 10.46121/pspc.54.1.24

12. M. Sahu, R. Dash, S. K. Mishra, M. Humayun, M. Alfayad and M. Assiri, "A deep transfer learning model for green environment security analysis in smart city," Journal of King Saud University-Computer and Information Sciences, vol. 36, no. 1, p. 101921, 2024.
13. X. a. W. L. Zhao, Y. Zhang, X. Han, M. Deveci and M. Parmar, "A review of convolutional neural networks in computer vision," vol. 57, no. 4, p. 99, 2024.
14. G. Van Houdt, C. Mosquera and G. N{\'a}poles, "A review on the long short-term memory model," vol. 53, no. 8, pp. 5929--5955, 2020.
15. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja and H. Chen, "Harris hawks optimization: Algorithm and applications," Future Generation Computer Systems, vol. 97, pp. 849-872, 2019.
16. K. Prathapchandran and T. Janani, "A trust aware security mechanism to detect sinkhole attack in RPL-based IoT environment using random forest--RFTRUST," Computer Networks, vol. 198, p. 108413.
17. D. Sharma, I. Mishra and S. Jain, "A detailed classification of routing attacks against RPL in internet of things," International Journal of Advance Research, Ideas and Innovations in Technology, vol. 3, no. 1, pp. 692--703, 2017.
18. F.-H. Tseng, L.-D. Chou and H.-C. Chao, "A survey of black hole attacks in wireless mobile ad hoc networks," Human-centric Computing and Information Sciences, vol. 1, pp. 1--16, 2011.
19. F. S. d. Lima Filho, F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar and L. F. Silveira, "Smart detection: an online approach for DoS/DDoS attack detection using machine learning," Security and Communication Networks, vol. 2019, no. 1, p. 1574749, 2019.
20. Y. Zong and G. Huang, "A feature dimension reduction technology for predicting DDoS intrusion behavior in multimedia internet of things," Multimedia Tools and Applications, vol. 80, no. 15, pp. 22671--22684, 2021.
21. R. Radhika and K. Kulothungan, "Mitigation of Distributed Denial of Service Attacks on the Internet of Things," Appl. Math, vol. 13, no. 5, pp. 831--837, 2019.
22. M. A. Bouke, A. Abdullah, S. H. ALshatebi, M. T. Abdullah and H. El Atigh, "An intelligent DDoS attack detection tree-based model using Gini index feature selection method," Microprocessors and Microsystems, vol. 98, p. 104823, 2023.
23. A. a. K. R. a. Z. S. Mustapha, F. Chbib, A. Fadlallah, W. Fahs and A. El Attar, "Detecting DDoS attacks using adversarial neural network," Computers \& Security, vol. 127, p. 103117, 2023.
24. S. A. Khanday, H. Fatima and N. Rakesh, "Implementation of intrusion detection model for DDoS attacks in Lightweight IoT Networks," Expert Systems with Applications, vol. 215, p. 119330, 2023.
25. S. J. Rani, I. Ioannou, P. Nagaradjane, C. Christophorou, V. Vassiliou, S. Charan, S. Prakash, N. Parekh and A. Pitsillides, "Detection of DDoS attacks in D2D communications using machine learning approach," Computer Communications, vol. 198, pp. 32--51, 2023.
26. M. K. a. H. A. A. a. I. S. a. S. N. a. A. S. N. H. S. a. P. B. Hasan, "DDoS: Distributed denial of service attack in communication standard vulnerabilities in smart grid applications and cyber security with recent developments," Energy Reports, vol. 9, pp. 1318--1326, 2023.

DOI: 10.46121/pspc.54.1.24

27. U. Shahid, M. Zunnurain Hussain, M. Zulkifl Hasan, A. Haider, J. Ali and J. Altaf, "Hybrid Intrusion Detection System for RPL IoT Networks Using Machine Learning and Deep Learning," IEEE Access, vol. 12, pp. 113099-113112, 2024.
28. E. Omar, S. Eman, M. Mohamed and E. Karim, "EIDM: deep learning model for IoT intrusion detection systems," Journal of Supercomputing, vol. 79, p. 13241–13261, 2023.
29. W. Xiao, D. Lie and Y. Guang, "A network intrusion detection system based on deep learning in the IoT," vol. 80, p. 24520–24558, 2024.
30. K. Yesi Novaria, N. Siti, S. Deris and Y. S. Bhakti, "An end-to-end intrusion detection system with IoT dataset using deep learning with unsupervised feature extraction," International Journal of Information Security, p. 1619–1648, 2024.
31. A. Fatani, M. Abd Elaziz, A. Dahou, M. A. A. Al-Qaness and S. Lu, "IoT Intrusion Detection System Using Deep Learning and Enhanced Transient Search Optimization," IEEE Access, vol. 9, pp. 123448-123464, 2021.
32. A. Dawoud, O. A. Sianaki, S. Shahristani and C. Raun, "Internet of Things Intrusion Detection: A Deep Learning Approach," in 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp. 1516-1522.
33. T. Stefanos, L. Thomas and R. Konstantinos, "Deep Learning in IoT Intrusion Detection," Journal of Network and Systems Management, vol. 30, 2022.

DOI: 10.46121/pspc.54.1.24