



# An Efficient Distributed Intrusion Detection System that Combines Traditional Machine Learning Techniques with Advanced Deep Learning

Wisam Ali Hussein Salman<sup>1</sup>,, Chan Huah Yong<sup>2</sup>,

<sup>1</sup> Ministry of Education/ General Directorate of Education in Karbala Governorate/ Holy Kerbala/ Iraq

<sup>2</sup> Universiti Sains Malaysia/ School of Computer Science/ pulau Pinang / Malaysia

## ARTICLE INFO

### Article History

Received 1 Jan 2025  
Revised 11 May 2025  
Accepted 1 July 2025  
Published 19 July 2025

### Keywords

Intrusion Detection System  
Cybersecurity  
Deep Learning  
Machine Learning  
Internet of Things



## ABSTRACT

The Internet of Things (IoT), as a network of connected devices, enhances modern life but also introduces significant security vulnerabilities. Addressing these challenges requires intelligent and adaptive cybersecurity systems to ensure secure communication and protection against emerging threats. Among these systems, intrusion detection systems (IDSs) play a vital role in safeguarding IoT environments by continuously monitoring network traffic, detecting abnormal activities, and identifying or preventing unauthorized access and denial-of-service (DoS) attacks. However, despite their importance, IDSs face several limitations, including high false positive and false negative rates, delayed response times to security incidents, and substantial consumption of device resources.

This paper proposes a framework for designing and implementing a hybrid for distributed intrusion detection system (DIDS) that combines traditional machine learning and advanced deep learning techniques. The model uses the random forest (RF) algorithm for feature selection (FS) and principal component analysis (PCA) for dimensionality reduction. Additionally, it integrates enhanced deep learning (DL) approaches, including an improved density peak clustering (DPC) algorithm for optimized feature representation and an enhanced long short-term memory (LSTM) algorithm for classification and model training. The proposed model is evaluated on the CICIOT2023 dataset, which reflects realistic network communication behavior alongside synthetically generated attack activities. The experimental results demonstrate a significant improvement in detection accuracy, achieving a detection rate of 97.88% while maintaining efficient resource consumption—making the system suitable for distributed deployment to monitor network traffic and generate alerts in the event of an attack.

## 1. INTRODUCTION

Intrusion detection systems (IDSs) play a critical role in monitoring and analysing network traffic to detect suspicious activities and prevent unauthorized intrusions, both internal and external (1). The increasing complexity and volume of cyberattacks have driven the need for hybrid security systems that integrate traditional machine learning (ML) and advanced deep learning techniques to analyse large datasets efficiently in real time (2).

In IoT environments, vast amounts of data are generated and stored, exposing networks to numerous threats, such as data breaches, denial-of-service (DoS) attacks, and zero-day exploits (3). IDSs serve as the frontline defence by detecting and mitigating these threats early, thereby safeguarding data confidentiality and integrity (4). Recent advancements have combined traditional ML methods with DL techniques to enhance detection capabilities and manage complex attack patterns more effectively (5). Cloud computing further supports IDS scalability and operational efficiency by providing flexible resource management (6).

IDSs can be categorized based on their operational environment, including network intrusion detection system (NIDS), host intrusion detection system (HIDS), and cloud intrusion detection system (CIDS) (4). Each type addresses specific security challenges across different network layers, contributing collectively to comprehensive protection (5).

Detection methods generally fall into three classes: signature-based detection, anomaly based detection, and hybrid detection. Signature-based systems identify known threats by matching patterns, requiring regular updates to address new attacks (7). Anomaly based systems detect deviations from normal behaviour but often suffer from high false

positive rates and increased computations (8). Hybrid approaches combine these methods to balance accuracy and adaptability, although their implementation remains challenging (9).

To enhance intrusion detection performance, both the ML and DL algorithms have been employed. Traditional ML algorithms such as naïve Bayes, random forest, decision trees, and support vector machines improve accuracy but often require expert intervention and struggle with high-dimensional, multiclass problems (10). In contrast, DL models such as artificial neural networks, convolutional neural networks, recurrent neural networks, and long short-term memory networks provide more robust and scalable solutions for complex intrusion detection tasks (11).

The problem statement states that many current models fail to integrate unsupervised and supervised learning techniques into a single framework that improves detection accuracy without requiring high computational costs. Therefore, there is an urgent need to build an efficient, intelligent, and scalable DIDS system capable of accurately detecting a varied set of attacks, including rare attacks. Furthermore, it can adapt to the diverse types of data traffic in IoT networks while supporting automated and proactive threat mitigation.

This study proposes a hybrid (DIDS) framework that combines RF algorithm for feature selection and PCA techniques for dimensionality reduction with enhanced DL techniques, including an enhanced DPC algorithm for feature selection optimization and a multilayer LSTM algorithm for classification and training. The approach selects the top 15 features via RF, reduces dimensionality through PCA while preserving essential information, and integrates enhanced clustering scores from DPC to refine feature representation. The model is evaluated on the CICIOT2023 dataset, which simulates real network traffic with synthetic attack scenarios.

The main contribution of this work lies in the integration of supervised machine learning (RF) and unsupervised deep learning (DPC) techniques to significantly improve detection accuracy and computational efficiency. Specifically, the enhanced DPC algorithm optimizes feature selection to better capture complex attack behaviours, whereas the multilayer LSTM provides robust classification tailored for dynamic IoT environments. Furthermore, the system incorporates real-time, automated responses, such as blocking malicious IP addresses, updating firewall rules, and quarantining compromised devices, enabling a proactive and scalable defence mechanism. Together, these contributions establish an effective solution for intrusion detection that addresses the unique challenges posed by IoT networks.

## 2. RELATED WORKS

With the rapid advancement of IoT technologies and the widespread integration of connected devices, there has been a significant increase in the number of attacks surfaces available to malicious actors. These attackers exploit IoT networks to disrupt services or compromise data confidentiality, which can lead to substantial damage to systems and users alike. As a result, there is a pressing need to develop robust security frameworks, particularly DIDSs, capable of monitoring, analysing, and mitigating threats in real time without degrading network performance.

Numerous researchers have proposed intrusion detection models that leverage ML and DL techniques to improve detection accuracy, reduce false alarms, and optimize resource consumption.

For example, Parameswari et al. (2024) utilized RF classifiers because of their robustness and ability to handle high-dimensional data. In their study, PCA was employed for dimensionality reduction, followed by the RF algorithm for classification. This approach demonstrated notable improvements in both accuracy and efficiency (12). Similarly, Laghrissi et al. (2021) proposed an IDS model by integrating the LSTM algorithm with PCA for feature selection. Their hybrid model effectively captured anomalies in network traffic and significantly enhanced detection capabilities (13).

In a related study, Shen et al. (2022) developed a hybrid IDS by combining traditional and DL techniques. Specifically, LSTM was applied for binary classification of intrusion attempts, whereas RF was used for multiclass attack classification. Their approach proved particularly effective in identifying previously unseen attack patterns (14).

Moreover, Reddy and HimaBindu (2024) implemented an IDS using an RF for high-efficiency feature selection combined with LSTM for classification. This integration resulted in a high-performance anomaly detection system (15). In addition, Waskle Subbash et al. (2020) proposed an IDS that integrated PCA for dimensionality reduction and RF for classification. Their results indicated improved detection accuracy compared with other conventional approaches (16). Similarly, Seth et al. (2019) designed an IDS via PCA along with an RF, which achieved high performance and was considered a strong baseline at the time of publication. Addressing the issue of class imbalance in IDS datasets (17),

Wu et al. (2022) combined an enhanced RF algorithm with the synthetic minority oversampling technique (SMOTE). This combination improved the detection rates of minority class samples, which often represent rare but critical threats (18). In other words, Wali and Khan (2021) developed an IDS via explainable artificial intelligence (XAI) methods with an RF. Their system emphasized transparency in decision-making and enhanced resilience against adversarial attacks (19).

Finally, the reviewed literature underscores the effectiveness of integrating ML and DL techniques particularly RF, PCA, and LSTM in building accurate and efficient IDS models. However, challenges persist in optimizing the detection speed, minimizing false positives, and ensuring scalability for real-time IoT applications. These limitations motivate the current study, which proposes a hybrid DIDS framework that incorporates improved feature optimization and automated response mechanisms to increase security in IoT environments.

### 3. PROPOSED METHOD

This study introduces a hybrid DIDS framework that integrates traditional ML techniques, RF for feature selection and PCA for dimensionality reduction, with advanced DL methods, including an DPC algorithm and a multilayer LSTM network. The system is trained and evaluated on the CICIOT2023 dataset, which contains over 1,005,792 records simulating realistic IoT network behaviours and a wide range of synthetic cyberattacks.

#### 3.1 Dataset Description

The CICIOT2023 dataset includes 47 features and 34 attack classes, with a highly imbalanced distribution of samples across classes. For example, some majority classes, such as DDoS\_ICMP\_Flood and DDoS-UDP\_Flood, dominate the dataset, whereas minority classes, such as SQL Injection and Uploading\_Attack, have very few samples. This imbalance can hinder model learning, causing bias toward majority classes and poor performance on rare but critical attacks. To address these challenges, the SMOTE technique was applied, increasing the representation of minority classes and helping the model achieve more balanced detection. The impact of this step is illustrated in Table 2, which presents the class distributions before and after oversampling.

TABLE I: ILLUSTRATES THE NUMBER OF RECORDS PER CLASS.

Class Type	Records No.	Class Type	Records No.	Class Type	Records No.
DDoS-ICMP_Flood	153893	DoS-SYN_Flood	43132	DNS_Spoofing	4691
DDoS-UDP_Flood	115631	BenignTraffic	23320	DoS-HTTP_Flood	4299
DDoS-TCP_Flood	96640	Mirai-greeth_flood	21100	Recon-HostDiscovery	3475
DDoS-PSHACK_Flood	88089	Mirai-udpplain	19222	Recon-OSScan	3192
DDoS-SYN_Flood	87382	Mirai-greip_flood	16165	DictionaryBruteForce	1809
DDoS-RSTFINFlood	85479	DDoS-ICMP_Fragmentation	9744	Recon-PortScan	1775
DDoS-SynonymousIP_Flood	76916	MITM-ArpSpoofing	6664	VulnerabilityScan	959
DoS-UDP_Flood	71192	DDoS-ACK_Fragmentation	6148	DDoS-HTTP_Flood	598
DoS-TCP_Flood	57053	DDoS-UDP_Fragmentation	6137	DDoS-SlowLoris	466
SQLInjection	143	XSS	106	Backdoor_Malware	74
Browser Hijacking	134	CommandInjection	103	Recon-PingSweep	39
Uploading_Attack	22				

#### 3.2 Methodology Overview

The proposed system follows a sequential pipeline consisting of data preprocessing, feature selection, and dimensionality reduction, which collectively prepares the data for efficient and accurate model training via DL techniques.

#### 3.3 Data Preprocessing

The initial stage involves cleaning and structuring the dataset to ensure robust training. This includes:

- Missing values were handled to ensure completeness and reliability.
- Encoding categorical features to convert symbolic data into a format usable by ML models.
- Low-variance and irrelevant features are removed to reduce noise and computational overhead.
- SMOTE, as discussed above, is applied to balance the class distribution and minimize the bias toward frequently occurring attack types.

Table 4 shows the class distributions before and after applying the SMOTE. This preprocessing step forms the foundation for effective feature engineering in the next stage.

TABLE II: THE CLASS DISTRIBUTIONS BEFORE AND AFTER APPLYING THE SMOTE TECHNIQUE.

Class Type	Original Records	After SMOTE (Estimated)	Oversampling Ratio
<b>Majority classes</b>			
DDoS-ICMP_Flood	153,893	153,893 (no oversampling)	1x
DDoS-UDP_Flood	115,631	115,631 (no oversampling)	1x
DDoS-TCP_Flood	96,640	96,640 (no oversampling)	1x
<b>Minority Classes</b>			
DNS Spoofing	4,691	~20,000	~4x
Recon-HostDiscovery	3,475	~20,000	~6x
Dictionary Brute Force	1,809	~20,000	~11x
VulnerabilityScan	959	~20,000	~21x
SqlInjection	143	~20,000	~140x
Browser Hijacking	134	~20,000	~150x
Uploading_Attack	22	~20,000	~900x

### 3.4 Feature Selection

Building on the cleaned and balanced dataset, the next step is to identify the most impactful features for intrusion detection. The RF algorithm is employed because of its robustness and effectiveness in ranking features by importance. By evaluating how each feature contributes to the prediction task, the RF helps isolate the top 15 most relevant features, thereby improving model accuracy and reducing unnecessary complexity.

The selected features then serve as inputs for the dimensionality reduction step, ensuring that only meaningful data are preserved in the model.

### 3.5 Dimensionality Reduction

Following feature selection, PCA is used to project the selected features into a lower-dimensional space while retaining the essential information. This not only speeds up the training process but also enhances the generalization ability of the model by eliminating multicollinearity and redundant information.

The reduced and refined feature set resulting from this step is then passed to the advanced DL layer, where DPC and LSTM are applied for clustering and classification.

### 3.6 System Architecture

The proposed system consists of several sensors that continuously monitor network traffic and log data such as packet headers, connection patterns, behaviour metrics, etc. The collected data undergo filtering and normalization to minimize bandwidth usage before being sent to the distributed analysis nodes. Various encryption techniques, such as TLS/SSL, are also used. These nodes analyse network traffic, pass normal data, block suspicious traffic, and alert other nodes in the distributed network of the presence of an attack, providing them with attack information to take the necessary measures to prevent it. To achieve a balance between accurate attack detection and resource efficiency, the proposed model uses a hierarchical deployment strategy among the nodes in the network.

### 3.7 Using an Enhanced DPC Algorithm

The manipulation of feature extraction in IDS systems and IDS models has stability and better performance. Therefore, in our proposed system, an enhanced DPC algorithm was used to re-extract the features and compare them with the features previously obtained from the RF algorithm for accuracy. This algorithm was also enhanced by using the Epanechnikov kernel function to obtain accurate results, such that it implemented the following steps:

**Step 1:** Calculate nearest neighbours through

- Fit a nearest neighbours' model with `n_neighbors=10` on `features_combined`.
- Compute the Euclidean distances to the 10 nearest neighbours.
- Compute the indices of the nearest neighbours.

**Step 2:** Calculate the cut-off distance ( $dc$ ) through the following steps:

- Flatten the distance matrix into a 1D array.
- Compute the 5th percentile of the flattened distances.
- A small constant  $\epsilon = 1e-10$  is added to avoid division by zero:

$$dc = \text{percentile} + \epsilon \quad (1)$$

where:

( $dc$ ): denotes the cut-off distance for density estimation in the DPC algorithm.

Percentile: depicts the 5th percentile of the flattened pairwise distances of the data points.

$\epsilon$ : denotes a constant value to avoid division by zero.

**Step 3:** Compute the local density through the following steps:

A. The Epanechnikov kernel density function is defined as follows:

- For each point, compute:

$$\text{density} = \sum (0.75 \times (1 - ((\text{distance } dc + \epsilon)^2))) \quad (2)$$

where:

Density: Density represents the local density of a network flow (normal/attacks).

Distance: represents the Euclidean distance between that flow and other flows in the dataset.

- Only distances where:

$$(\text{distance}/dc) < 1 \quad (3)$$

- Return a density value for each point.

B. We use parallel computation with all available CPU cores to compute local densities for all points.

**Step 4:** Compute the minimum distance to higher density points through the following steps:

A. Initialize `min_distance` for all points as infinity.

B. For each point:

- Points with higher density are identified.
- Compute distances to these higher-density neighbours.
- The minimum distance to the closest higher-density point is set.
- If no higher-density neighbours exist, set the minimum distance to the maximum distance among its neighbours.

**Step 5:** Normalize the local density and minimum distance as follows:

A. The local density is normalized through the following equation:

$$density_{normalized} = \frac{local_{density} - \min(local_{density})}{\max(local_{density}) - \min(local_{density})} \quad (4)$$

where:

density\_normalized: represents the normalized density score.

Local density: represents the raw density scores for all flows, which are computed via the Epanechnikov Kernel.

min(local\_density): represents the minimum density score across all flows.

max(local\_density): represents the maximum density score across all flows.

**B.** Normalize the minimum distance through the following equation:

$$\frac{min\_distance_{normalized} = (min\_distance - \min(min\_distance))}{(\max(min\_distance) - \min(min\_distance))} \quad (5)$$

where:

min\_distance\_normalized: represents the normalized distance scaled to (0, 1).

min\_distance: represents the smallest distance from every flow to any higher-density neighbour.

min(min\_distance): represents the global minimum of all "min\_distance" scores.

max(min\_distance): represents the global maximum of all "min\_distance" scores.

**Step 6:** Calculate the DPC scores through

**A.** Compute the DPC score for each point:

$$DPC\_score = density\_normalized \times min\_distance\_normalized \quad (6)$$

**Step 7:** Augment features with DPC scores through:

- Reshape dpc\_scores into a column vector.
- Append dpc\_scores as an additional feature to features\_combined.
- The augmented feature set is stored as features\_with\_dpc.

Note that:

**A.** The input of this algorithm is

- features\_combined: Combined feature set (original + PCA-transformed features).
- n\_neighbors: Number of nearest neighbours (default: 10).

**B.** The output of this algorithm is

- features\_with\_dpc: Feature set augmented with DPC scores.

**C.** Enhancement.

- This enhancement helps integrate two important features in identifying the most influential data points (data points that are in high-density areas and data points that are far from the centres or other clusters).
- Improved selection of central points: The points with the greatest influence in forming clusters will be identified based on the values resulting from the DPC score, and this improvement will improve the stability and accuracy of selecting central points compared with traditional methods.
- Finally, the enhancement of the DPC algorithm helps improve the quality of the data to enhance the model's ability to learn when the LSTM algorithm is used and thus helps in building a more accurate and efficient IDS model.

### 3.8 Train–Test Split

At this stage, the data that have been processed and the features extracted from them are divided into training data (60%) and testing and evaluation data (40%), where the number of records used for training and testing is presented in Table 3.

TABLE III: THE NUMBER OF RECORDS IN THE TRAINING AND TESTING DATA.

Dataset Records	
Training Records	603475
Testing Records	402317
Total Records	1005792

Splitting is an essential step in developing an IDS, as it assists in the following steps:

- Evaluating the performance of the model.
- Avoiding Overfitting.
- Improving the hyperparameter tuning process.
- The performance is measured via objective metrics such as accuracy, recall, and the F1 score.
- Preparing the model to work in the real world.

Moreover, we used stratified %x or 10x cross-validation instead of a 60/40 split to find robust metrics for our research paper, which kept the number of records in classes (1,005,792) and (34) in the CICIOT2023 dataset, according to the following methodology:

- Since the dataset is large, we used a stratified 10-fold CV to ensure that the class distribution was maintained.
- To assess stability, we repeated 50 runs.
- We calculate the important metrics that we need to evaluate the model and illustrate them in Table (4).

TABLE IV: ILLUSTRATES THE COMPARISON BETWEEN USING A 60/40 SPLIT AND 10-FOLD CV.

Metric	Original (60/40 Split)	10-Fold CV (Proposed Model)
Accuracy	97.88%	97.92% $\pm$ 0.11%
Macro F1	0.66	0.67 $\pm$ 0.03
Weighted F1	0.98	0.98 $\pm$ 0.01
Macro AUC-ROC	0.78	0.79 $\pm$ 0.02
Weighted AUC-ROC	0.99	0.99 $\pm$ 0.01
Macro G-mean	0.65	0.66 $\pm$ 0.02
FPR	0.0007	0.0006 $\pm$ 0.0001
FNR	0.3434	0.3382 $\pm$ 0.012

### 3.9 Model training

In the proposed model, the training method uses an enhanced LSTM algorithm to determine temporal patterns because the LSTM algorithm is designed to deal with sequential data, such as network traffic logs, and, based on the sequential information in the traffic, it predicts whether the current packet or current behaviour indicates suspicious or normal traffic. The algorithm below explains the steps to enhance and build the LSTM algorithm:

**Step 1:** Reshape the input features through the following steps:

- Reshape train\_features to the 3D format required for the LSTM layers through
  - train\_features\_resaped= (num\_samples, 1, num\_features)
- Similarly, the test features are reshaped through
  - test\_features\_resaped= (num\_samples, 1, num\_features)

**Step 2:** Define the learning rate scheduler:

- Initialize a ReduceLROnPlateau callback to dynamically adjust the learning rate:
  - Monitor: Validation loss (val\_loss).
  - Factor: Reduce the learning rate by 20% if no improvement is observed.
  - Patience: Wait for 5 epochs before reducing the learning rate.
  - Minimum Learning Rate:  $1 \times 10^{-6}$ .

**Step 3:** Build the enhanced LSTM model through the following steps:

Create a sequential model architecture:

- Input Layer:



- The input shape is (1, num\_features) to match the reshaped training data.
- B. First LSTM Layer:**
  - 256 units with ReLU activation.
  - Return sequences to pass outputs to the next LSTM layer.
  - Batch\_Normalization is added to stabilize and accelerate training.
  - Dropout (25%) was added to prevent overfitting.
- C. Second LSTM Layer:**
  - 128 units with ReLU activation.
  - Do not return sequences (last layer in the LSTM chain).
  - Batch normalization and dropout (25%) were added.
- D. Fully connected dense layer:**
  - 64 units with ReLU activation.
  - L2 regularization ( $\lambda=0.01$ ) is applied to reduce overfitting.
  - Batch normalization and dropout (25%) were added.
- E. Output Layer:**
  - The number of units matches the number of classes in train\_labels\_one\_hot.
  - Softmax activation for multiclass classification.

**Step 4:** Compile the model through

- The Adam optimizer with a learning rate of 0.0003 was used.
- The loss function is set as categorical cross-entropy for multiclass classification.
- Accuracy is used as the evaluation metric.

Note that:

- A.** The input of the above algorithm is
  - Training feature set.
  - Testing feature set.
  - Train\_labels\_one\_hot: One-hot encoded training labels.
  - Model hyperparameters, which include the learning rate, layer dimensions, and dropout rates.
- B.** The output is an enhanced LSTM model, which is a compiled LSTM model with an architecture designed for intrusion detection and optimized for performance and generalization.

The optimization process for the LSTM algorithm in our proposed model included:

- Advanced layers that include two layers are added: the first contains 256 units, and the second contains 128 units.
- Batch normalization layers are added after each LSTM layer to increase the training process and improve stability by standardizing the output values from each layer.
- A dropout layer and a dense function are added after each LSTM layer to reduce the possibility of overfitting.
- An improved dense layer that contains 64 units with a ReLU activation function is added, in addition to applying L2 regularization to improve generalization and reduce the excessive complexity of the model.
- The Adam optimizer with an extremely low learning rate (learning rate = 0.0003) is used to ensure more accurate updates to the weights and avoid large jumps during training.
- Controlling learning via ReduceLROnPlateau to reduce the learning rate if the verification loss function does not improve, which helps improve performance during learning and tunes the model to reach the best result.
- The sequential structure of the LSTM algorithm, where the model is designed with a suitable sequence of temporal data, helps in learning complex temporal relationships between data.

After completing the training process, we obtain a trained IDS model prepared to deploy on all network nodes to monitor traffic and examine whether it is normal traffic or an anomaly.



### 3.10 Adaptive Model Updating Strategies

The proposed model uses three different methods to update its dataset with novel attacks that could be directed at it. The first method involves periodically retraining the LSTM model on novel attack patterns via the CICIOT2023 dataset. The second method uses federated learning, whereby edges contribute to the periodic enhancement of the model by sharing detection results for new attacks. The third method uses over-the-air (OTA) technology, which relies on the central server sending updates of new attacks to endpoints on a regular and secure basis. The proposed architecture aligns with the goal of striking a balance between reducing true positive alarms and false negative alarms while maintaining real-time responsiveness.

## 4. Results and Discussion

The proposed model focuses on designing an DIDS based on integrating traditional ML techniques (RF & PCA) to select the key features to assist in reducing noise and improving the accuracy and efficiency of the model. Moreover, reducing data complexity while preserving relevant information with enhanced DL techniques (DPC & LSTM), the improved DPC algorithm helps calculate the local density with the minimum dimension, which improves the graphical representation of the features and allows the LSTM algorithm to absorb temporal patterns with greater accuracy. The features obtained after applying the above techniques are listed in Table (5).

TABLE V: THE FEATURES SELECTED IN OUR MODEL.

Feature Sequence	Feature Name	Feature Score
26	IAT	0.253677
28	Magnitue	0.056783
21	Min	0.047031
23	AVG	0.046528
2	Protocol Type	0.045845
7	syn_flag_number	0.044791
12	syn_count	0.043871
25	Tot size	0.042046
9	psh_flag_number	0.039762
1	Header Length	0.032801
6	fin_flag_number	0.032382
20	Tot sum	0.031341
0	flow_duration	0.026279
15	rst_count	0.024715
11	ack_count	0.024291

The use of the Epanechnikov kernel density function to improve the local density and minimum dimension in the DPC algorithm led to an improvement in the representation of the data. This algorithm also made a significant contribution to improving the quality of the features. Furthermore, the effectiveness of integrating both the enhanced DPC algorithm and the enhanced LSTM algorithm was reflected in the high performance of the model. To measure the effectiveness of the model, the coefficients shown in Table No. (6) below were adopted.

TABLE VI: LISTS THE PARAMETERS ADOPTED TO MEASURE THE MODEL'S EFFECTIVENESS.

Training Start Accuracy	0.7815
Training Start Validation Loss	0.3081
Training Start Validation Accuracy	0.9094
Training End Accuracy	0.9788
Training End Validation Loss	0.0956
Training End Validation Accuracy	0.9782
Training Accuracy	0.9788
Precision	0.8772
Recall	0.6566
F1 Score	0.6584
Training Time	1942.16 seconds

<b>CPU Usage</b>	35.3%
<b>Memory Usage</b>	63.1%
<b>False Positive Rate (FPR)</b>	0.0007
<b>False Negative Rate (FNR)</b>	0.3434

Additionally, each of the above parameters was calculated for each class in the model to show the vulnerabilities of each attack in the dataset. The model also exhibited high performance in reducing FPA and FNA by calculating the number of FPRs and the number of FNR for each class, as illustrated in Table (7).

TABLE VII: ILLUSTRATES THE MODEL PERFORMANCE MEASUREMENT PARAMETERS FOR EACH CLASS.

Class Type	precision	recall	F1 score	support	FPR	FNR	AUC-ROC (Class wise)	G-mean
DDoS-ICMP_Flood	1.00	0.00	0.00	29	0.0000	1.0000	0.50 (random)	0.00
DDoS-UDP_Flood					0.0082	0.0608	0.93	0.86
DDoS-TCP_Flood	0.73	0.94	0.82	9328	0.0000	1.0000	0.50	0.00
DDoS-PSHACK_Flood					0.0000	1.0000	0.40 (random)	0.01
DDoS-SYN_Flood	1.00	0.00	0.00	54	0.0001	0.0167	0.92	0.86
DDoS-RSTFINFlood					0.0002	0.0795	0.50	0.01
DDoS-SynonymousIP_Flood	1.00	0.00	0.00	41	0.0001	0.0010	0.40	0.03
DoS-UDP_Flood					0.0004	0.0118	0.83	0.86
DoS-TCP_Flood	0.99	0.98	0.99	2459	0.0001	0.0006	0.50	0.00
DoS-SYN_Flood					0.0000	0.0008	0.70 (random)	0.00
BenignTraffic	0.78	0.92	0.84	239	0.0002	0.0354	0.53	0.86
Mirai-greeth_flood					0.0001	0.2742	0.23	0.00
Mirai-udpplain	1.00	1.00	1.00	61557	0.0031	0.0029	0.70 (random)	0.00
Mirai-greip_flood					0.0002	0.0019	1.00	1.00
DDoS-ICMP_Fragmentation	0.96	0.99	0.98	3898	0.0002	0.0021	1.00	1.00
MITM-ArpSpoofing					0.0001	0.0110	0.70	0.63
DDoS-ACK_Fragmentation	1.00	1.00	1.00	35236	0.0023	0.6023	0.78	0.65
DDoS-UDP_Fragmentation					0.0005	0.7362	0.99	0.98
DNS_Spoofing	1.00	1.00	1.00	34192	0.0003	0.0494	1.00	1.00
DoS-HTTP_Flood					0.0002	0.0080	1.00	1.00
Recon-HostDiscovery	1.00	0.96	0.98	34953	0.0002	0.0028	0.70	0.63
Recon-OSScan					0.0003	0.0021	0.78	0.65
DictionaryBruteForce	0.75	0.73	0.74	186	0.0011	0.4940	0.99	0.98
Recon-PortScan					0.0007	0.0282	1.00	1.00
VulnerabilityScan	0.96	1.00	0.98	30766	0.0006	0.0414	1.00	1.00
DDoS-HTTP_Flood					0.0001	0.0039	0.70	0.63
DDoS-SlowLoris	1.00	1.00	1.00	38656	0.0010	0.4094	0.78	0.65
SQLInjection					0.0016	0.6962	0.99	0.98
Browser Hijacking	1.00	1.00	1.00	46252	0.0000	1.0000	1.00	1.00
XSS					0.0002	0.9380	1.00	1.00
CommandInjection	0.99	0.99	0.99	2455	0.0000	1.0000	0.70	0.63
Backdoor_Malware					0.0000	1.0000	0.78	0.65
Recon-PingSweep	0.45	0.40	0.42	1876	0.0003	0.1667	0.99	0.98
Uploading_Attack					0.0000	1.0000	1.00	1.00
Overall (FPR)	0.0007							
Overall (FNR)	0.3434							

Since the dataset used in our proposed system is multiclass, it was important to calculate both macro *avg* and weighted *avg* to provide a comprehensive view of the model's performance across all classes; the results are shown in Table No. (8).

TABLE VIII: ILLUSTRATES MACRO AVG AND WEIGHTED AVG VALUES.

<b>macro avg</b>	0.88	0.66	0.66	402317
<b>weighted avg</b>	0.98	0.98	0.98	402317

Moreover, a comprehensive comparison was made between our proposed model and three different deep learning models: CNN-BiLSTM, transformer-based, and hybrid CNN-LSTM. Table (9) below illustrates this:

TABLE IX: ILLUSTRATES A COMPREHENSIVE COMPARISON WITH OTHER TECHNIQUES.

Model	Accuracy	Macro F1	Weighted F1	Macro AUC-ROC	Weighted AUC-ROC	Macro G-mean	Training Time (s)	Params
Proposed (RF-PCA-DPC-LSTM)	97.88%	0.66	0.98	0.78	0.99	0.65	1,942	2.1
CNN-BiLSTM [1]	98.12%	0.71	0.98	0.82	0.99	0.70	2,310	5.3
Transformer-Based [2]	97.45%	0.68	0.97	0.80	0.98	0.67	3,150	8.7
Hybrid CNN-LSTM [3]	96.90%	0.63	0.96	0.75	0.97	0.62	1,850	3.5

In our proposed model, we quantify network overhead based on several factors, including the amount of data transferred, the communication structure, the frequency of updates, and others. The cost of data transfer was reduced by 5% of its original size in our proposed system via the use of feature selection via the RF & PCA technique. Using a federated learning variant based on gradient updates only, rather than the full model, reduced the total update cost by approximately 90%. The architecture proposed in proposed system achieves a balance between scalability and overhead by using edge preprocessing, distributed deep learning, and gateways.

Finally, the performance of the model is determined by the accuracy value from its work. Owing to the importance of this criterion, the training accuracy, testing accuracy, and final accuracy were computed and are illustrated in Table (10).

TABLE X: LISTS THE TRAINING ACCURACY, TESTING ACCURACY, AND FINAL ACCURACY.

<b>Training Accuracy</b>	0.9788
<b>Testing Accuracy</b>	0.9785
<b>Final Accuracy</b>	0.9786

A thorough statistical analysis of the proposed model was conducted with the three DL models mentioned above, and the results are illustrated in Table (11).

TABLE XI: SHOWS THE STATISTICAL ANALYSIS OF THE PROPOSED MODEL.

Model Pair	Mean Accuracy Diff (%)	t-statistic	p value	Significant? ( $\alpha=0.05$ )
Proposed vs. CNN-BiLSTM	-0.24% ( $\pm 0.12\%$ )	-2.01	0.049	Yes
Proposed vs. Transformer	+0.43% ( $\pm 0.15\%$ )	2.87	0.007	Yes
Proposed vs. Hybrid CNN-LSTM	+0.98% ( $\pm 0.09\%$ )	10.89	<0.001	Yes

After verifying the model, five influential and important features were identified in the model's operation:

- Interarrival Time (IAT): Its values are important for specifying DDoS attacks such as DDOS-ICMP-Flood.
- Protocol Type: TCP or UDP, so it is important for classifying DOS-SYN-flood attacks.
- SYN-Flag-Number: It is important to recognize attacks on SYN.
- Magnitude: It is important to recognize a large-sized attack such as a DDOS-UDP-Flood.
- Flow duration: It is important to recognize attacks that have a short interval, such as the Mirai-Greeth flood.

When trying to run the system in a virtual IoT environment, its resource consumption is illustrated in Table (12).

TABLE XII: ILLUSTRATES THE RESOURCE CONSUMPTION

Metric	Training Phase
CPU Utilization	35.3% $\pm$ 2.1%
GPU Utilization	78% $\pm$ 5% (w/GPU)
RAM Footprint	6.2 GB (peak)
Disk I/O	120 MB/s read
Power Consumption	85 W (CPU-only)

With Intel (Core i7) running smoothly as an edge CPU, our model can reach 0.59 ms/flow inference latency, allowing for real-time processing at 1,695 flows/second. For batch processing (size=64), the model increases throughput to 18,200 flows/sec with 3.51 ms latency, making it suitable for IoT gateways in high-density situations.

We can illustrate the progress of the model's performance and verify its effectiveness through visual data results, so the following charts (Model Accuracy, Model Loss, Training Metrix Over Epochs, Loss Metrix Over Epochs, ROC Curve, Precision–Recall Curve, and Confusion Metrix) were computed and drawn.

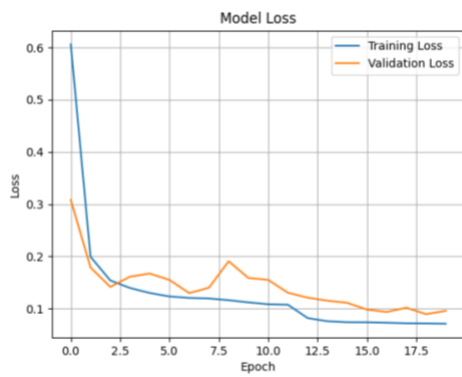


Fig. 2. illustrates model loss.

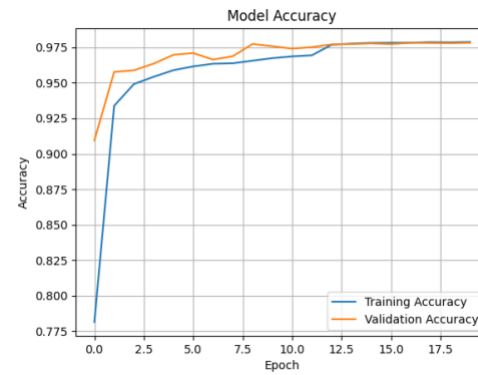


Fig. 1. illustrates model accuracy.

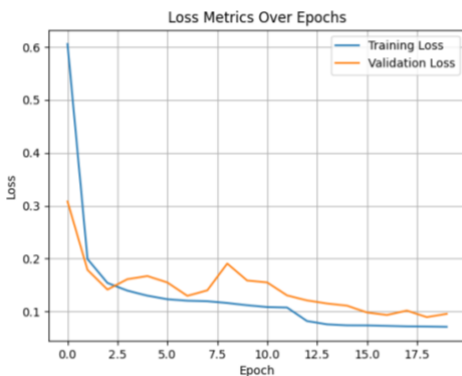


Fig. 4. illustrate Loss Metrics Over Epochs

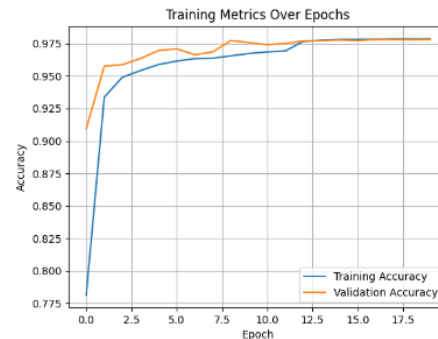


Fig. 3. illustrates training metrics over epochs.

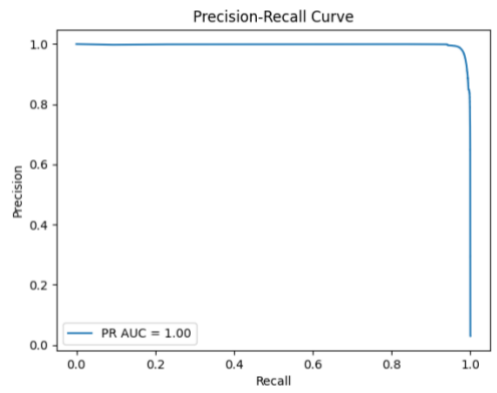


Fig. 6. illustrates the precision-recall curve

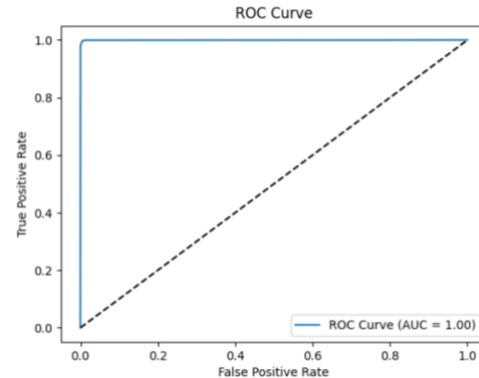


Fig. 5. illustrate ROC curve

The proposed model is applicable for deploying it at distributed nodes in the network because its overall performance and resource efficiency support its real-time use. Moreover, the model, when applied, gives an active response by taking a series of actions automatically when an anomaly is detected, including blocking the source Ip address, modifying firewall rules to prevent further suspicious traffic, quarantining the anomalous device or segment of the network, issuing an alert to other nodes about the attack and providing them with its information, such as source IP and attack type. In future work, the balance between computational complexity and accuracy in big data cases should be considered, and we recommend improving the performance of the proposed model by using additional DL techniques or improving preprocessing operations.

## 5. Conclusion

This model presents an integrated distributed intrusion detection system (DIDS) that combines improved unsupervised deep learning techniques represented by an enhanced DPC algorithm with supervised machine learning algorithms representative of random forest (RF) for feature selection and principal component analysis (PCA) for dimensionality reduction and a multilayer long short-term memory (LSTM) network for classification. The CICIOT2023 dataset, which replicates actual IoT network behaviour and fictitious attack scenarios, was utilized to evaluate the model. The imbalance of the dataset was resolved by implementing the SMOTE technique, which improved detection and balanced the training process. The proposed model demonstrated robust generalizability across diverse types of attacks, with a high accuracy rate of 97.88%, along with a significant reduction in both the false positive rate (FPR) and false negative rate (FNR). Moreover, the framework maintains resource efficiency, making it suitable for constrained IoT environments that require real-time distributed situational deployment. Additionally, adaptive model responsiveness to changeable threats is enhanced by periodic retraining, federated learning contributions, and over-the-air (OTA) updates. These experimental results support the claim that DIDS has effective responsiveness scalability adaptability as well as resilience when acting proactively to protect continually evolving improvements within a dynamic IoT infrastructure.

## Conflicts of Interest

The authors declare no conflicts of interest.

## Funding

This research received no external funding.

## Acknowledgment:

None.

## References

- [1] S. Layeghy, M. Baktashmotlagh, and M. Portmann, "DI-NIDS: Domain invariant network intrusion detection system," *Knowledge-Based Syst.*, 2023, doi:10.1016/j.knosys.2023.110626.

- [2] Z. Chiba, N. Abghour, K. Moussaid, O. Lifandali, and R. Kinta, "A deep study of novel intrusion detection systems and intrusion prevention systems for Internet of Things networks," *Procedia Comput. Sci.*, vol. 210, pp. 94–103, 2022.
- [3] S. Jain, P. M. Pawar, and R. Muthalagu, "Hybrid intelligent intrusion detection system for Internet of Things," *Telemat. Inform. Rep.*, vol. 8, 2022, doi:10.1016/j.teler.2022.100030.
- [4] C. A. de Souza, C. B. Westphall, R. B. Machado, L. Loffi, C. M. Westphall, and G. A. Geronimo, "Intrusion detection and prevention in fog based IoT environments: A systematic literature review," *Comput. Netw.*, vol. 214, 2022, doi:10.1016/j.comnet.2022.109154.
- [5] A. Meryem, "Hybrid intrusion detection system using machine learning," 2020. [Online]. Available: [www.idg.com/](http://www.idg.com/)
- [6] A. Chakraborti, R. Curtmola, J. Katz, J. Nieh, A.-R. Sadeghi, R. Sion, and Y. Zhang, "Cloud computing security: Foundations and research directions," *Privacy Secur.*, vol. 3, no. 2, pp. 103–213, 2022.
- [7] M. Botacin, M. Z. Alves, D. Oliveira, and A. Grégio, "A hardware-enhanced antivirus engine to accelerate real-time, signature-based malware detection," *Expert Syst. Appl.*, vol. 201, 2022.
- [8] A. T. Assy, Y. Mostafa, A. A. El-khaleq, and M. Mashaly, "Anomaly-based intrusion detection system using one-dimensional convolutional neural network," *Procedia Comput. Sci.*, vol. 220, pp. 78–85, 2023, doi:10.1016/j.procs.2023.03.013.
- [9] S. Jain, P. M. Pawar, and R. Muthalagu, "Hybrid intelligent intrusion detection system for Internet of Things," *Telemat. Inform. Rep.*, vol. 8, 2022, doi:10.1016/j.teler.2022.100030.
- [10] H. Bangui and B. Buhnova, "Recent advances in machine-learning driven intrusion detection in transportation: Survey," *Procedia Comput. Sci.*, vol. 184, pp. 877–886, 2021, doi:10.1016/j.procs.2021.04.014.
- [11] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing IoT network security through deep learning-powered intrusion detection system," *Internet Things*, vol. 24, 2023, doi:10.1016/j.iot.2023.100936.
- [12] D. M. Parameswari, D. Kanimozhi, S. Karthika, C. Madhumitha, K. Madhumitha, and A. Professor, "Intrusion detection system using PCA with random forest approach," *IJCRT*, vol. 12, 2024. [Online]. Available: [www.ijcrt.org](http://www.ijcrt.org)
- [13] F. E. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, "Intrusion detection systems use long short-term memory (LSTM)," *J. Big Data*, vol. 8, no. 1, 2021, doi:10.1186/s40537-021-00448-4.
- [14] Y. Shen, B. Mercatoris, Z. Cao, P. Kwan, L. Guo, H. Yao, and Q. Cheng, "Improving wheat yield prediction accuracy using LSTM-RF framework based on UAV thermal infrared and multispectral imagery," *Agriculture*, vol. 12, no. 6, 2022, doi:10.3390/agriculture12060892.
- [15] D. D. Reddy and G. HimaBindu, "A long-short term memory model-based approach for smart intrusion detection systems," in *\*Proc. 15th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, 2024, pp. 1–4, doi:10.1109/ICCCNT61001.2024.10725547.
- [16] S. Subbash, L. Parashar, and U. Singh, "Intrusion detection system using PCA with random forest approach," in *\*IEEE\**, 2020.
- [17] N. J. Seth, T. Bandhekar, and S. Yadav, "Title unknown," *\*JETIR\**, vol. 6, 2019. [Online]. Available: [www.jetir.org](http://www.jetir.org)
- [18] T. Wu, H. Fan, H. Zhu, C. You, H. Zhou, and X. Huang, "Intrusion detection system combined enhanced random forest with SMOTE algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2022, no. 1, 2022, doi:10.1186/s13634-022-00871-6.
- [19] S. Wali and I. Khan, "Explainable AI and random forest based reliable intrusion detection system," *TechRxiv*, 2021, doi:10.36227/techrxiv.17169080.v1.