A Lightweight MLP-Based Feature Extraction with Linear Classifier for Intrusion Detection System in Internet of Things

Jisi Chandroth and Jehad Ali *

Posted Date: 25 March 2026

doi: 10.20944/preprints202603.1915.v1

Keywords: internet of things; intrusion detection system; multi-layer perceptron; deep learning

 doi:10.20944/preprints202603.1915.v1

Article

A Lightweight MLP-Based Feature Extraction with Linear Classifier for Intrusion Detection System in Internet of Things

Jisi Chandroth [1] and Jehad Ali [2,*]

[1] Department of AI and Software, Gachon University, Seongnam 13120, South Korea
[2] Department of AI Convergence Network, Ajou University, Suwon, 16499, South Korea
* Correspondence: jehadali@ajou.ac.kr

Abstract

The Internet of Things (IoT) comprises diverse devices connected through heterogeneous communication protocols to deliver a wide range of services. However, the complexity and scale of IoT networks make them difficult to secure. Network intrusion detection systems (NIDS) have therefore become essential for identifying malicious activities and protecting IoT environments across many applications. Although recent deep learning (DL)-based IDS approaches achieve strong detection performance, they often require substantial computation and storage, which limits their practicality on resource-constrained IoT devices. To balance detection accuracy with computational efficiency, we propose a lightweight deep learning model for IoT intrusion detection. Specifically, our method learns compact, intrusion-relevant representations from traffic features using a two-layer Multi-Layer Perceptron (MLP) embedding backbone, followed by a linear SoftMax classification head for multi-class attack detection. We evaluate the proposed approach on two benchmark datasets, CICIDS2017 and NSL-KDD, and the results show strong performance, achieving 99.85% and 99.21% accuracy, respectively, while significantly reducing model size and computational overhead. Experimental results demonstrate that the proposed method achieves excellent classification performance while maintaining a lightweight design, with fewer parameters and lower FLOPs than existing approaches.

Keywords: internet of things; intrusion detection system; multi-layer perceptron; deep learning

1. Introduction

The Internet of Things (IoT) is considered to be among the rapidly developing technological spheres in the world. It helps in merging the physical space with the cyberspace. The IoT technology is a network consisting of interrelated physical objects, sensors, actuators, and others that communicate through different communication protocols to create, receive, and transmit data. It has become popular in various fields of application, such as the healthcare, the transport and industrial systems, to enhance the overall quality of experience in everyday life [1].

Although the IoT is widely applied in various uses, it has a low resistance against broad security threats. Heterogeneous and dispersed nature of the IoT devices, in addition to the lack of computational capabilities and security settings, render these environments nice targets by the cyber attackers. Therefore, IoT networks are frequently subject to numerous types of attacks, e.g., Distributed Denial of Service (DDoS) attacks, data losses, malware distribution, and unauthorized access. These security concerns have been underlined by the number of large-scale IoT attacks that have been observed in the last ten years. To illustrate, the malware campaign of BADBOX 2.0 has affected over one million devices by March 2025 and it is estimated that it might have reached over ten million devices in 222 countries by now [2]. Another notable one is the Reaper botnet which was identified in September 2017, unlike the Mirai botnet which primarily used weak default credentials, it used known vulnerabilities on IoT devices, making it more developed and threatening to use.

 doi:10.20944/preprints202603.1915.v1

Intrusion Detection System (IDS) is essential to alleviate these threats and to secure the safety of the IoT systems. IDS is an alternative that continuously inspects and analyzes network traffic and system activity to detect anomalies and possible intrusions [4]. It allows identifying different attacks in IoT networks. Over the last several years, a significant portion of the attention has been focused on IDS methods that rely on Machine Learning (ML) and Deep Learning (DL) since they can conduct automatic learning of complex patterns of the network traffic data and enhance the accuracy of the detection process in the IoT contexts [5,6].

The IoT ecosystem consists of a wide range of devices, such as sensors, actuators and edge nodes, which are usually constrained by harsh resource limits. IoT devices usually possess weak processing power, memory, storage capacity, and energy sources than traditional computing systems do [7]. These limitations ensure that it is difficult to deploy traditional security mechanisms and computationally expensive intrusion detection systems to directly run on IoT devices. This has left most IoT devices vulnerable to security threats and exploitable weaknesses. Moreover, conventional IDS techniques usually depend on sophisticated machine learning algorithms that demand massive training data sets and high dimension features which demand a lot of computation and storage resources that cannot be implemented in the IoT setting [8].

The application of successful IDS solutions to IoT networks consequently poses a number of challenges. Currently, numerous methods focus primarily on high detection accuracy at the cost of constraints in the practical implementation of IoT devices, such as limited processing power, reliance on batteries, and limited memory. Moreover, IoT networks are large-scale and heterogeneous, making real-time intrusion detection even more difficult [9]. This may still lead to latency increase and extra energy usage in spite of the fact that edge computing has been implemented to remove computational workload of the IoT devices to nearby edge nodes [10,11]. Thus, when creating intrusion detection systems in an IoT setting, computational overhead, energy efficiency, and real-time detection are to be given due attention.

In effort to solve these problems, recent studies have aimed at coming up with lightweight IDS models that have been optimized to meet the needs of resource-constrained IoT settings. The main aim is to achieve the balance between high detection and low computational cost characteristics of high detection performance is sought after [12]. Nevertheless, the problem of creating machine learning-based IDS models, which are efficient and accurate, is a continuous challenge. In the recent past, researchers have investigated methods like compressed model and federated learning and dynamic quantization to simplify the models without significantly impacting the level of detection performance [13]. These methods are designed to facilitate the ability of effective intrusion detection within the constrained computational and energy capabilities of the IoT devices.

Although the advances have been achieved in the intrusion detection methods, most of the available IDS solutions to the IoT settings continue to not find the optimal balance between detection efficiency and computational efficiency. The majority of the traditional ML and DL-based methods are based on the complicated architecture and numerous parameters and consequently consume a lot of memory, computation time, and energy [14]. These features render them inappropriate to run on resource limited IoT devices. Hence, there is a strong necessity of light and efficient intrusion detection mechanisms which ensure that the detection performance is high with minimum consumption of resources. In this paper, we will develop an effective IDS that suits IoT settings and is optimized by balancing the detection rate with the performance rate. The proposed solution will aim at providing efficient intrusion detection and will be appropriate to be implemented in resource-constrained IoT systems.

The main contributions of this work are summarized as follows:

- We propose a lightweight intrusion detection model that learns compact representations from network traffic features using a two-layer Multi-Layer Perceptron embedding backbone.
- The proposed architecture employs a simple, efficient design that combines an MLP feature-embedding network with a linear SoftMax classification head for multi-class attack detection.

 doi:10.20944/preprints202603.1915.v1

- The proposed approach is evaluated on two widely used benchmark datasets, namely CICIDS2017 and NSL-KDD, to validate its effectiveness for intrusion detection tasks.
- Experimental results demonstrate that the proposed model achieves strong classification performance, with accuracies of 99.85% on CICIDS2017 and 99.21% on NSL-KDD.
- The proposed method maintains a lightweight structure with reduced model size, fewer parameters, and significantly lower FLOPs compared with existing approaches, making it suitable for deployment in resource-constrained IoT environments.

The rest of the paper will be structured in the following. Section 2 explains the lightweight methods of intrusion detection of the IoT networks. Section 3 presents the proposed intrusion detection model which is developed on the basis of two-layered Multi-Layer Perceptron embedding backbone and linear SoftMax classification head to detect multi-class attacks. Section 4 is the description of the dataset, experimental set-up, model implementation, evaluation metrics, and performance analysis. Lastly, Section 5 will wrap up the paper, comment on the findings, and provide the future work directions.

2. Related Works

As a result of the resource constraints of IoT devices, the recent studies have started focusing more on creating lightweight intrusion detection systems (IDSs) capable of performing effectively with limited resources in terms of computational power, memory, and energy. A number of lightweight IDS models have been proposed in order to meet these challenges. Nevertheless, most of the current techniques have not been able to deliver high accuracy on detection and minimize the complexity of the model. As an example, Z. Wang *et al.* [10] developed a lightweight IoT intrusion detection model that is built on an architecture of a DL-BiLSTM that uses deep neural networks (DNN) and bidirectional long short-term memory (BiLSTM) to learn nonlinear relationships and long-term temporal dependencies among network traffic. Incremental principal component analysis (IPCA) is employed to minimise the computational expenses incurred during dimensionality reduction of features, whereas post-training dynamic quantization is employed to minimise the computational costs incurred during model compression. Even though such a strategy decreases the computational cost, dynamic quantization can negatively impact detection accuracy compared to the original model and requires extra processing steps. Besides, the assessment dataset is not entirely representative of the complexity and dynamic nature of the actual world environment of IoT network, which reduces the generalization ability of the model.

In a bid to make lightweight IDS perform better, a number of studies have explored lightweight IDS techniques including feature optimization, knowledge distillation, and lightweight architectural design. According to Wang *et al.* [13], the model proposed is a lightweight intrusion detection model that uses self-knowledge distillation (SKD) and is known as the tied block convolution lightweight neural network (TBCLNN). In this model, binary Harris Hawk Optimization (bHHO) will be used to reduce the number of features and lightweight convolutional structures with residual and inverted residual blocks will be used to reduce the computation complexity. In a similar way, Benaddi *et al.* [15] presented a hybrid IDS model that uses convolutional neural networks (CNN) and BiLSTM networks to be useful in capturing both spatial and temporal characteristics in IoT traffic data. The UNSW-NB15 dataset is analyzed using a chi-square type feature selection algorithm to determine the most significant features of the dataset and the input dimensions are reduced to enhance the efficiency of the method.

Moreover, the recent work has investigated knowledge distillation and lightweight architectures to trade of detection performance and computational cost. As one such example, the CL-SKD framework [16] uses a two-step approach to learning that incorporates both self-supervised contrastive learning and self-knowledge distillation to improve representation learning and minimize the use of labeled data. The model first acquires traffic representations through contrastive learning, and

 doi:10.20944/preprints202603.1915.v1

subsequently learns directly through the teacher model to transfer the knowledge to a lightweight student model to enhance the performance on detecting.

Moreover, LNet-SKD [17] introduces a lightweight intrusion detection model using self-knowledge distillation. It presents a DeepMax block of effectively deriving compact traffic representations and piles up several of these blocks to build a lightweight model. It also includes batch-wise self-knowledge distillation to reduce the performance reduction due to simplification of the model. In the same manner, the inverted residual network combined with a multi-batch self-knowledge distillation mechanism to detect network intrusion is proposed in IRNet-MBSKD [18]. The inverted residual design that is protocol-aware promotes effective feature extraction, and the self-distillation strategy with many batches promotes generalization of the model. The strategy attains the high-detection rates on the NSL-KDD dataset and also consumes lower computational costs.

Several other studies have also discussed the optimization-based feature selection and advanced learning strategies to enhance effectiveness of lightweight IDSs. A deep learning-based intrusion detection model, DP2, proposed by Khan *et al.* [19] is an extension of DNN and Bi-LSTM-based intrusion detection with a wrapper-based genetic algorithm (GA) feature selection approach, which aims at eliminating redundant features and reducing memory consumption. Likewise, Yang *et al.* [20] have created a lightweight open-source framework that can be used to detect intrusion into industrial IoT or IIoT called a CompM3 that consists of known attack classification, unknown attack detection based on reconstruction error analysis, and dynamically updating to adapt to new attack patterns known to the system. Moreover, Ma *et al.* [21] proposed a cloud-edge-node architecture in which computationally intensive training is done in the cloud. Simultaneously, the lightweight detection modules are implemented on the edges and node tiers. Despite these methods lessening computational pressure at the device level, they also have a number of issues such as being vulnerable to adversarial attacks, dependent on particular malicious traffic patterns, and limited attention to post-detection mitigation measures.

Lightweight model compression, transfer learning and frequency-domain representations of features are also discussed in recent works as the way to detect intrusion into IoT. Zhang *et al.* [22] suggested a few-shot intrusion detection method based on lightweight transfer learning (NID-LTL) to combine model pruning, nonlinear feature selection, and knowledge distillation to generate a small-scale model that can be adapted to new attacks. Fard *et al.* [23] explored the design space optimization methods, and it allowed them to create smaller deep-neural-networks that fit on embedded IoT devices without significantly deteriorating the training process. Equally, Wang *et al.* [24] presented a lightweight CNN model, which used features of Fourier transforms with knowledge distillation to enhance features representation and generalizations. PNet-IDS [25] as well as other packet-embedding-based classification methods [26] and attention-enhanced BiLSTM models [27] are designed to achieve greater detection accuracy with lower model complexity.

However, even the majority of the existing strategies continue to face the problem of the optimal balance between the accuracy of the detection, the complexity of the model, and the possibility of real-time deployment. Most models are based on extra optimization methods, including quantization, pruning, or knowledge distillation, that may add extra computation or may even worsen detection. Moreover, the datasets applicable in numerous studies also fail to capture the highly dynamic and heterogeneous nature of a real-world IoT environment, which constrains the extrapolability of the models suggested. Problems like the imbalance of the dataset, the scalability of the large-scale network, the resistance to the changing attack pattern, etc. are not explicitly addressed, as well. Thus, it remains necessary to create lightweight intrusion detection models that can learn compact and useful traffic representations at the same time as with high detection rates and low computation costs, so that they can be suitable to be deployed easily in real-world IoT systems. Table 1 provides the summary of existing literature.

 doi:10.20944/preprints202603.1915.v1

Table 1. Comparison of Existing Lightweight IoT Intrusion Detection Methods

Ref.	Feature Selection	Model	Disadvantages
[10]	Incremental PCA (IPCA)	DNN + BiLSTM	• Dynamic quantization reduces the optimal detection capability of the trained model. • The dataset does not represent highly dynamic IoT environments.
[13]	Binary Harris Hawk Optimization (bHHO)	TBCLNN with SKD	• Performance strongly depends on the optimization process. • Complex lightweight convolution structures increase model design complexity.
[15]	Chi-square (χ^2)	CNN + BiLSTM	• Limited scalability in large IoT environments. • Reduced robustness under dynamic network conditions. • Dataset imbalance affects detection performance.
[19]	Genetic Algorithm (GA)	DNN + BiLSTM	• Quantization introduces accuracy degradation. • The dataset does not fully represent real IoT traffic variability.
[20]	–	VAE + Extreme Value Machine	• The EVM classifier produces unstable accuracy and F1-score. • Removing the EVT component significantly degrades detection performance.
[21]	CDF ranking + Gini importance	MLP, CNN, LSTM, RF	• Detection accuracy decreases when the number of selected features is reduced. • Performance depends heavily on the selected classifier.
[22]	Kernel-based nonlinear feature selection	VGG-based model	• Heavy dependence on pretrained models reduces adaptability to new traffic distributions. • Transfer learning increases training complexity.
[23]	–	DNN	• Design space exploration requires extensive computational search. • Model tuning significantly increases development time.
[24]	–	CNN with Knowledge Distillation	• The method focuses mainly on intrusion detection rather than attack family classification. • Simple oversampling does not effectively solve dataset imbalance.
[25]	–	Lightweight CNN (PNet-IDS)	• Performance degrades in large-scale IoT environments. • Model struggles with complex traffic patterns.
[26]	Packet embeddings	1D CNN	• Lower predictive accuracy compared to deep models. • Requires centralized server processing for improved classification.
[27]	–	Attention-based BiLSTM	• The model suffers from scalability issues with large network traffic volumes. • High computational overhead for large datasets.
[16]	–	CL-SKD	• The model contains many hyperparameters that require manual tuning. • Hyperparameter selection increases training complexity.
[17]	–	LNet-SKD	• Lightweight architecture reduces feature representation capability. • Knowledge distillation introduces additional training overhead.
[18]	–	IRNet-MBSKD	• Protocol-aware architecture increases model complexity. • Multi-batch self-distillation increases training time.

 doi:10.20944/preprints202603.1915.v1

3. Proposed Method

This study proposes a lightweight intrusion detection model for resource-constrained IoT environments. The main objective of the proposed method is to achieve high detection accuracy while maintaining low computational complexity and a small model size. The proposed architecture employs a two-layer Multi-Layer Perceptron feature embedding network, followed by a linear SoftMax classification layer, to learn compact representations of network traffic features. The overall framework of the proposed IDS is illustrated in Figure 1. The framework consists of three main stages: 1) data preprocessing, 2) feature extraction, and 3) classification. In the data preprocessing stage, raw network traffic data are processed to ensure data quality and consistency prior to model training. In the feature extraction stage, the preprocessed feature vectors are fed into the MLP layers, which learn compact representations of the traffic patterns. Finally, in the classification stage, a linear SoftMax classifier maps the learned feature embeddings to multiple attack classes for final prediction. The mathematical notations used in this work are shown in Table 2

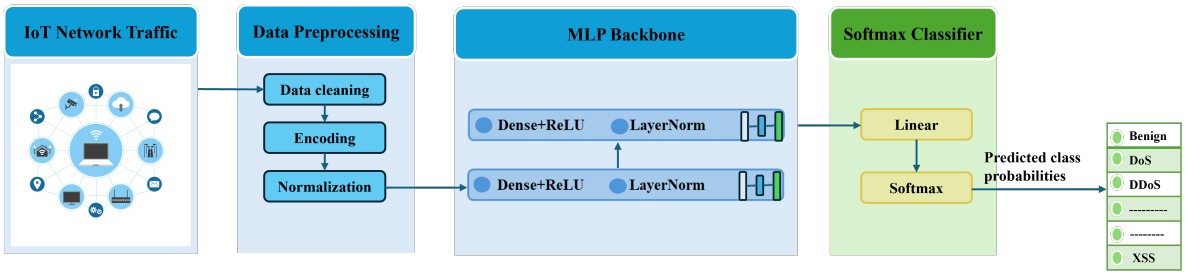


Figure 1. Overall system architecture.

3.1. Data Preprocessing

Let (X, Y) denote the original dataset, where $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{n \times d}$ represents the input feature matrix containing n samples and d features, and $Y = \{y_1, y_2, \ldots, y_c\}$ denotes the corresponding class labels. The label set Y comprises c traffic classes that represent different types of normal and malicious network activities. During data preprocessing, several operations are performed to ensure data quality and suitability for model training. First, missing values are removed to prevent potential bias and skewness in the learning process. Subsequently, rows containing non-finite values are filtered out to avoid numerical instability during model training. After cleaning the dataset, categorical class labels are converted to numerical values using *LabelEncoder*, which maps each class label to an integer in the range $\{0, 1, \ldots, c-1\}$.

To handle categorical attributes in the feature set, one-hot encoding is applied, transforming categorical variables into binary vectors, thereby making all features numerical. Let the resulting feature matrix after encoding be denoted as X_{enc}. Next, feature scaling is performed using Min–Max normalization to transform the feature values into a consistent range. Specifically, each feature value is normalized to the interval $[0, 1]$ using the following transformation:

$$X_{\text{norm}} = \frac{X_{\text{enc}} - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

where X_{min} and X_{max} represent the minimum and maximum values of each feature, respectively. The final preprocessed dataset can therefore be represented as (X_p, Y), where $X_p = X_{\text{norm}}$. This normalization step helps stabilize the training process and improves the convergence behavior of the neural network.

 doi:10.20944/preprints202603.1915.v1

Table 2. Mathematical notation used in the proposed model

Symbol	Description
n	Number of samples in the dataset
d	Number of input features
C	Number of traffic classes
X	Original feature matrix
Y	Class label vector
X_p	Preprocessed and normalized feature matrix
x_p	Input feature vector
h_1	Hidden representation of the first layer
z	Compact embedding vector from the second hidden layer
W_1	Weight matrix of the first hidden layer
b_1	Bias vector of the first hidden layer
W_2	Weight matrix of the second hidden layer
b_2	Bias vector of the second hidden layer
W_3	Weight matrix of the classification layer
b_3	Bias vector of the classification layer
$\sigma(\cdot)$	Activation function (ReLU)
$\hat{y}$	Predicted class probability vector
o	Output logits before SoftMax
L	Cross-entropy loss function

3.2. Feature Extraction

We employ a multi-layer perceptron for feature extraction. The MLP is a popular feedforward neural network architecture consisting of an input layer, one or more hidden layers, and an output layer. The network can learn intricate patterns in the data by using a nonlinear activation function after each neuron processes the input through weighted connections. In general, MLPs use forward propagation to generate predictions, backpropagation to update network parameters, and hidden-layer representation learning to extract meaningful patterns from input data.

After the preprocessing stage, the normalized feature matrix is represented as $X_p \in \mathbb{R}^{n \times d}$, where n denotes the number of samples and d represents the number of input features. Each input vector x_p is fed into the MLP through the input layer, where the number of input neurons is equal to the feature dimension of the dataset. The proposed model consists of an input layer followed by two hidden layers containing 128 and 64 neurons, respectively, and a final output layer for multi-class classification. The hidden layers employ the ReLU activation function to learn complex nonlinear patterns from IoT traffic data. In particular, the second hidden layer reduces the feature representation to 64 neurons, thereby generating a compact embedding representation of the network traffic features.

The first hidden layer projects the input vector into a higher-dimensional representation, which is defined as

$$h_1 = \sigma(W_1 x_p + b_1) \tag{2}$$

where $W_1 \in \mathbb{R}^{d \times h}$ and $b_1 \in \mathbb{R}^h$ denote the weight matrix and bias vector of the first layer, respectively, and h denotes the hidden layer dimension. Here, $\sigma(\cdot)$ represents the ReLU activation function, which introduces nonlinearity and allows the model to capture complex feature interactions in network traffic data.

The hidden representation is then transformed into a compact embedding vector through the second linear layer:

$$z = W_2 h_1 + b_2 \tag{3}$$

where $W_2 \in \mathbb{R}^{h \times e}$ and $b_2 \in \mathbb{R}^e$ denote the parameters of the second layer, and e represents the embedding dimension. The resulting vector $z \in \mathbb{R}^e$ serves as a low-dimensional representation of

 doi:10.20944/preprints202603.1915.v1

the input traffic features. This embedding vector captures the most relevant information required to distinguish between different network traffic categories. By compressing the input features into a compact representation, the embedding network reduces computational complexity while preserving discriminative characteristics necessary for accurate intrusion detection.

3.3. Classification Layer and Loss Function

After extracting the compact embedding representation $z \in \mathbb{R}^e$ from the feature embedding network, the embedding vector is passed to a linear SoftMax classification layer to predict the final traffic class. The classifier maps the embedding vector to the output space containing C traffic categories. The output logits are computed as:

$$\mathbf{o} = W_3 z + b_3 \tag{4}$$

where $W_3 \in \mathbb{R}^{e \times C}$ and $b_3 \in \mathbb{R}^C$ represent the weight matrix and bias vector of the classification layer, respectively, and C denotes the number of traffic classes in the dataset. The vector $\mathbf{o} \in \mathbb{R}^C$ contains the unnormalized prediction scores (logits) for each class.

To obtain the probability distribution over the classes, the SoftMax function is applied to the output logits:

$$\hat{y}_i = \frac{\exp(o_i)}{\sum_{j=1}^{C} \exp(o_j)}, \quad i = 1, 2, \ldots, C \tag{5}$$

where $\hat{y}_i$ represents the predicted probability for class i. The predicted class label is determined by selecting the class with the highest probability.

To train the model, the cross-entropy loss function is used to measure the difference between the predicted probabilities and the true class labels. The loss function is defined as

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{6}$$

where y_i represents the ground-truth label encoded in one-hot form and $\hat{y}_i$ denotes the predicted probability for class i. Minimizing this loss encourages the model to assign higher probabilities to the correct class labels. The pseudocode for the proposed algorithm is shown in Algorithm 1.

Algorithm 1 Proposed Lightweight MLP-Based Intrusion Detection Model

Require: Original dataset (X, Y), learning rate η, batch size B, number of epochs E
Ensure: Predicted class labels $\hat{y}$

```
Preprocess the data to obtain preprocessed dataset (X_p, Y)
Initialize weights W_1, W_2, W_3 and biases b_1, b_2, b_3
for epoch = 1 to E do
    for each mini-batch B ⊂ X_p of size B do
        for each input sample x_p ∈ B do
            Compute predicted probabilities ŷ using the MLP and SoftMax classifier
            Compute cross-entropy loss L
        end for
        Update model parameters using AdamW optimizer
    end for
end for
return predicted class labels ŷ
```

4. Experimental Setup and Performance Analysis

4.1. Datasets

In this section, we describe the datasets used to evaluate the performance of the proposed intrusion detection model. Two widely used benchmark datasets, CIC-IDS2017 [28] and NSL-KDD [29], are used

 doi:10.20944/preprints202603.1915.v1

to assess the effectiveness of the proposed approach across different network intrusion scenarios. The datasets were divided into 80% for training and 20% for testing to assess the generalization capability of the proposed approach. The training subset was used to learn the model parameters, while the testing subset was used to evaluate the final detection performance.

4.1.1. CICIDS2017

The CICIDS2017 is a large, realistic dataset commonly used to test Network Intrusion Detection Systems. The Canadian Institute of Cybersecurity (CIC) created it to address weaknesses in previous datasets on intrusion-detection systems. The datasets include normal network traffic and contemporary attack conditions that mirror real cyber threats. The data were collected in a controlled network environment over five working days (Monday-Friday) to simulate the normal activities of an organization and various forms of wrongdoing. The dataset includes various network traffic classes, including Benign, PortScan, Hulk, DDoS, FTP, Bot, and Web Attack, among others. Such types of attacks render CICIDS2017 appropriate for analyzing the resilience of intrusion detection frameworks in determining various forms of cyber-attacks.

4.1.2. NSL-KDD

The NSL-KDD dataset is an improved version of the earlier KDD'99 dataset and is widely used for benchmarking intrusion detection methods. It is designed to represent different types of network traffic behaviors, including both normal and malicious activities. The dataset consists of one normal traffic class and several attack categories such as neptune, satan, smurf, and portsweep. These attack categories provide a diverse set of intrusion patterns for evaluating the performance of intrusion detection models.

4.2. Experimental Setup

All experiments were implemented in Python using the PyTorch deep learning framework. Data preprocessing and analysis were performed using commonly used scientific computing libraries, including NumPy, Pandas, Matplotlib, and Seaborn. The experiments were conducted on a system equipped with an Intel Core i5-12600K processor running at 3.69 GHz with 48 GB RAM.

4.3. Model Training and Hyperparameter Settings

The proposed model employs a two-layer Multi-Layer Perceptron architecture with 128 and 64 neurons in the hidden layers, respectively. The hidden layers utilize the ReLU activation function to capture nonlinear relationships in network traffic data, while the output layer applies a SoftMax classifier for multi-class traffic classification. The model was trained for 100 epochs with a batch size of 128. Parameter optimization was performed using the AdamW optimizer with a learning rate of 0.003, and L2 regularization (weight decay of 1×10^{-4}) was applied to improve generalization and prevent overfitting. The hyperparameters used in the proposed model are summarized in Table 3.

Table 3. Hyperparameter settings of the proposed model

Parameter	**Value**
Hidden layer 1 neurons	128
Hidden layer 2 neurons	64
Activation function	ReLU
Optimizer	AdamW
Learning rate	0.003
Weight decay (L2 regularization)	1×10^{-4}
Batch size	128
Epochs	100
Loss function	Cross-entropy

 doi:10.20944/preprints202603.1915.v1

4.4. Evaluation Metrics

To evaluate the effectiveness of the proposed intrusion detection model, several standard performance metrics were employed, including Accuracy, Precision, Recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). These metrics provide a comprehensive assessment of the classification performance of the proposed model.

Accuracy measures the overall proportion of correctly classified samples among the total number of samples. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

where TP (True Positive) represents correctly detected attack samples, TN (True Negative) denotes correctly identified normal samples, FP (False Positive) refers to normal samples incorrectly classified as attacks, and FN (False Negative) represents attack samples incorrectly classified as normal traffic.

Precision measures the proportion of correctly predicted attack samples among all predicted attack samples and is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall, also known as detection rate, measures the proportion of actual attack samples that are correctly identified by the model:

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

The F1-score represents the harmonic mean of Precision and Recall and provides a balanced measure of the model performance:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{10}$$

In addition, the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is used to evaluate the ability of the model to distinguish between different classes across various threshold values. A higher AUC value indicates better classification performance.

4.5. performance Results

Tables 4 and 5 show the per-class performance of the proposed model. The metrics include accuracy, precision, recall, and F1-score. In Table 4, the proposed model performs very well across most classes in the CICIDS2017 dataset. The Benign, DDoS, FTP, Hulk, and PortScan classes show accuracy values above 99%. Their precision, recall, and F1-scores are also above 0.99. This shows strong classification performance. However, the Bot class shows lower performance. This indicates that the model sometimes confuses Bot traffic with benign traffic. The Web Attack class also shows slightly lower precision. However, the overall detection performance remains high. Table 5 shows the results for the NSL-KDD dataset. The proposed model performs well for most classes. The Neptune class achieves perfect precision of 1.00. It also shows a high F1-score of 0.9985. The Normal class also performs well. The PortSweep, Satan, and Smurf classes show slightly lower values. However, their accuracy and F1-scores remain high. These results show that the proposed model effectively detects different attack types on both datasets.

 doi:10.20944/preprints202603.1915.v1

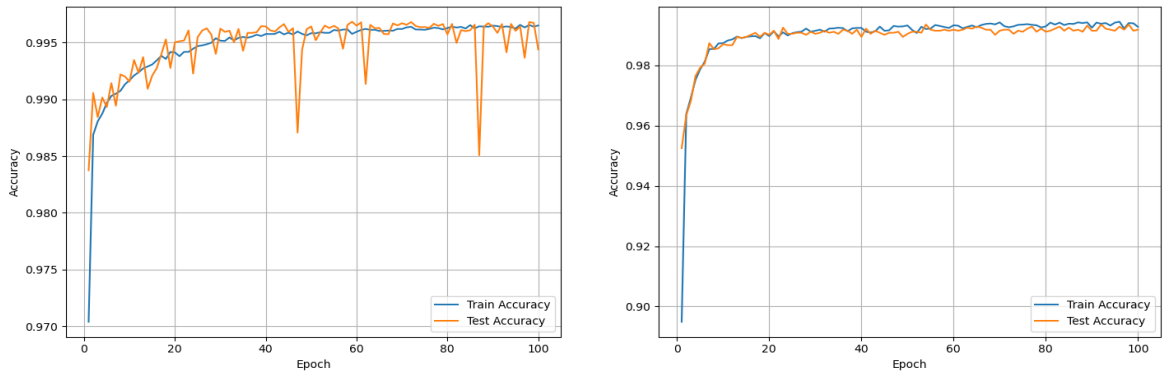
Table 4. Per-class performance metrics cicids2017

Class	Accuracy	Precision	Recall	F1-score
Benign	0.9971	0.9961	0.9971	0.9966
Bot	0.6814	1.0000	0.6814	0.8105
DDoS	0.9997	0.9994	0.9997	0.9995
FTP	0.9983	0.9983	0.9983	0.9983
Hulk	0.9989	0.9955	0.9989	0.9972
PortScan	0.9996	0.9992	0.9996	0.9994
Web Attack	0.9912	0.9825	0.9912	0.9868

Table 5. Per-class performance metrics nsl-kdd

Class	Accuracy	Precision	Recall	F1-score
Neptune	0.9970	1.0000	0.9970	0.9985
Normal	0.9911	0.9958	0.9911	0.9933
PortSweep	0.9643	0.9474	0.9643	0.9558
Satan	0.9905	0.9286	0.9905	0.9585
Smurf	0.9753	0.9405	0.9753	0.9576

The accuracy curves of the proposed model over the training epochs are illustrated in Figure 2 for the CICIDS2017 and NSL-KDD datasets. The plots show training and test accuracy values over 100 epochs. As shown in Figure 2(**a**), for the CICIDS2017 dataset, the training and testing accuracy increase rapidly during the initial epochs and gradually stabilize as training progresses. Although minor fluctuations in testing accuracy are observed at certain epochs, the training and testing curves remain closely aligned. Similarly, Figure 2(**b**) illustrates the training performance on the NSL-KDD dataset. The model converges quickly within the first few epochs and reaches maximum accuracy. The training and test accuracy curves closely overlap throughout training, indicating stable learning behavior.

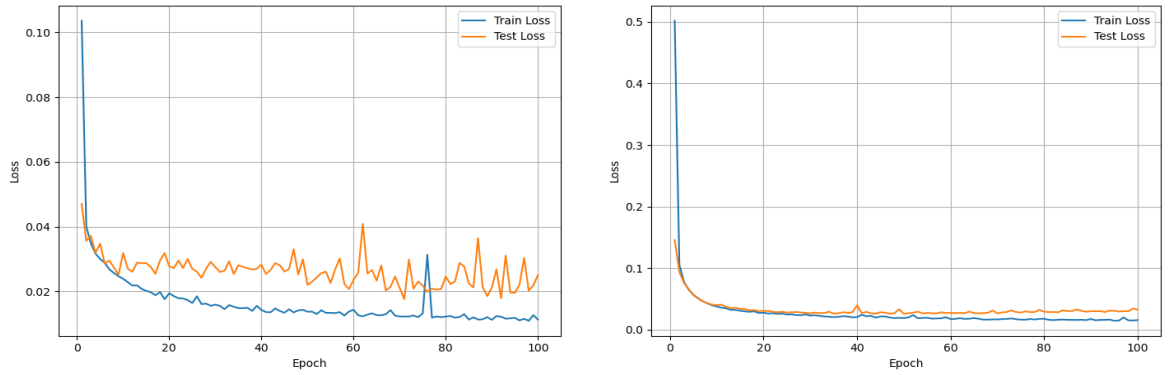


(**a**) CICIDS2017 (**b**) NSL-KDD

Figure 2. Accuracy curves of the proposed model in different datasets.

The loss curves of the proposed model across training epochs are illustrated in Figure 3 for the CICIDS2017 and NSL-KDD datasets. The plots show the training and testing losses over 100 epochs. As shown in Figure 3(**a**), for the CICIDS2017 dataset, the training loss decreases rapidly during the initial epochs and gradually stabilizes as the training progresses. Similarly, Figure 3(**b**) presents the loss curves for the NSL-KDD dataset. The training and test losses decrease sharply in the early epochs and converge quickly as training continues. Both curves remain closely aligned throughout the training process.

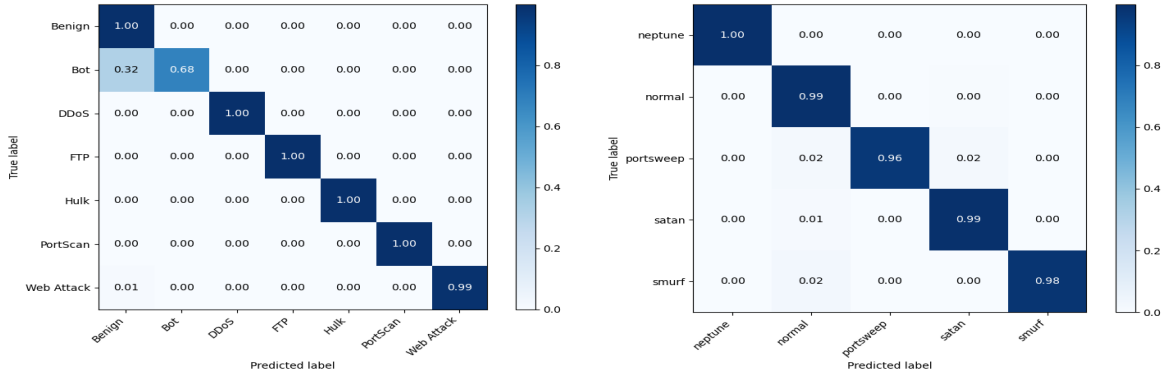
 doi:10.20944/preprints202603.1915.v1



(**a**) CICIDS2017 (**b**) NSL-KDD

Figure 3. Loss curves of the proposed model in different datasets.

The normalized confusion matrix shown in Figure 4 illustrates the relationship between the true and predicted class labels produced by the proposed model. The diagonal elements represent the correctly classified samples and therefore indicate the classification accuracy for each class, whereas the off-diagonal elements correspond to misclassified samples. By observing the diagonal elements across the two datasets, it can be inferred that the proposed model achieves high classification performance for most attack categories. However, the results also show that the model is confused when identifying some specific attack types. For instance, as shown in Figure 4(**a**), for the CIC-IDS2017 dataset, the proposed model misclassifies approximately 32% of the Bot attacks as Benign traffic, while the remaining classes are accurately classified. In contrast, Figure 4(**b**) shows the results for the NSL-KDD dataset, where the proposed model almost perfectly classifies each attack category with very few misclassifications.

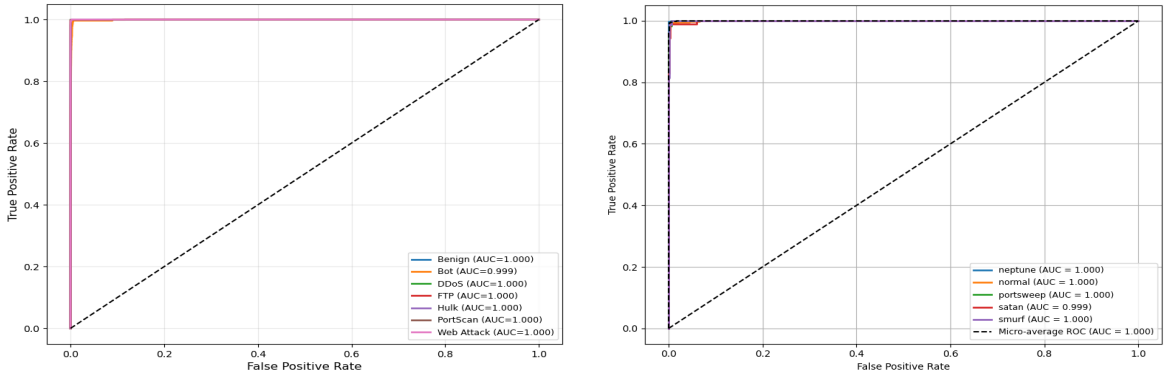


(**a**) CICIDS2017 (**b**) NSL-KDD

Figure 4. Confusion matrix of the proposed model in different datasets.

Figure 5 shows the ROC curves of the proposed model for the CICIDS2017 and NSL-KDD datasets. In Figure 5(**a**), the ROC curves for the CICIDS2017 dataset are presented. The curves are located near the top-left corner of the plot. This indicates strong classification performance. Most classes achieve AUC values close to 1.0. The Benign, DDoS, FTP, Hulk, PortScan, and Web Attack classes show perfect discrimination. The Bot class has a slightly lower AUC compared to the other classes. In Figure 5(**b**), the ROC curves for the NSL-KDD dataset are shown. The curves also remain very close to the top-left corner. This indicates strong detection capability for the attack classes. The Neptune, Normal, and Smurf classes achieve an AUC value of 1.0. The PortSweep and Satan classes show slightly lower values.

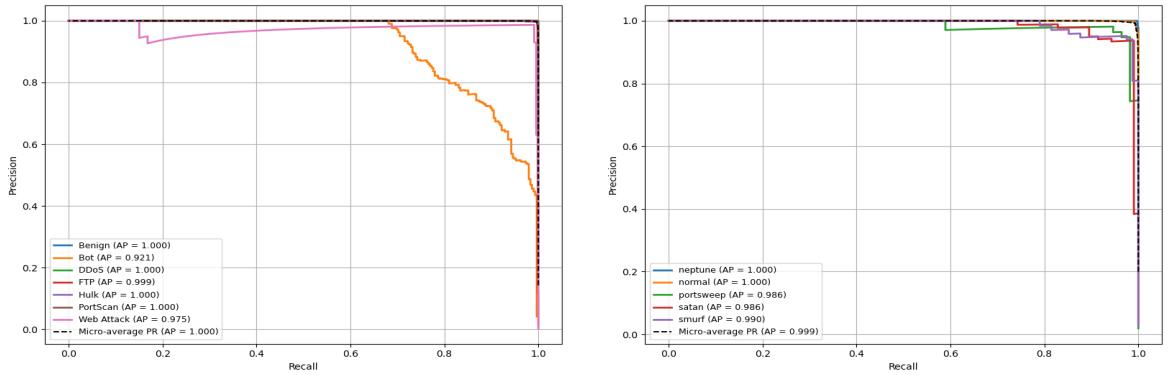
 doi:10.20944/preprints202603.1915.v1



(**a**) CICIDS2017 (**b**) NSL-KDD

Figure 5. ROC of the proposed model in different datasets.

Figure 6 shows the precision–recall curves of the proposed model for the CICIDS2017 and NSL-KDD datasets. In Figure 6(**a**), most classes show precision–recall curves close to the top-right corner. This indicates strong classification performance. The Benign, DDoS, FTP, Hulk, PortScan, and Web Attack classes achieve very high average precision values. The Bot class performs worse than the other classes. The curve for the Bot class decreases as recall increases, indicating that the model struggles to detect Bot attacks in some cases. In Figure 6(**b**), the curves remain close to the top-right corner for most classes, indicating high precision and recall. The Neptune, Normal, and Smurf classes show very high average precision values. The PortSweep and Satan classes show slightly lower values. However, their performance remains strong.



(**a**) CICIDS2017 (**b**) NSL-KDD

Figure 6. precision–recall curves of the proposed model in different datasets.

4.6. Computational Complexity Analysis

Table 6 presents the computational complexity and model size of the proposed intrusion detection method. The results show that the proposed model maintains a lightweight structure while achieving strong detection performance. For the CICIDS2017 dataset, the model requires 22,827 parameters with 45,200 FLOPs and 22,600 MAC operations, resulting in a model size of 89.6 KB. The training time is 26.98 minutes, and the testing time is 1.33 seconds. For the NSL-KDD dataset, the model uses 13,573 parameters, 26,752 FLOPs, and 13,376 MAC operations, resulting in a model size of 54.4 KB. The training time is 43.67 seconds, and the testing time is 0.05 seconds. These results show that the proposed model requires low computational resources and small storage space. Therefore, the model is suitable for deployment in resource-constrained IoT environments.

 doi:10.20944/preprints202603.1915.v1

Table 6. Computational complexity and model size of the proposed method

Dataset	Training time (s)	Testing time (s)	No. parameters	FLOPs	MACs	Model size (KB)
CICIDS2017	26.98 min	1.33	22827	45200	22600	89.6
NSL-KDD	43.67	0.05	13573	26752	13376	54.4

4.7. Comparison Study

Different existing lightweight intrusion detection models are used as comparison methods, including DL-BiLSTM [10], CL-SKD [16], LNet-SKD [17], and IRNet-MBSKD [18].

The performance comparison with existing lightweight intrusion detection models is shown in Table 7. The proposed model achieves strong classification performance on both datasets. For the CICIDS2017 dataset, the proposed method obtains an accuracy of 99.85% and an F1-score of 99.93%. These results are slightly higher than the compared methods. At the same time, the proposed model requires only 13,573 parameters and 26,752 FLOPs. This computational cost is significantly lower than several existing models.

For the NSL-KDD dataset, the proposed model achieves an accuracy of 99.21% and an F1-score of 99.22%. These results are higher than the other compared approaches. In addition, the computational complexity remains very low. The number of parameters and FLOPs is much smaller than the deep learning models reported in previous studies. These results demonstrate that the proposed lightweight architecture provides strong detection capability while maintaining low computational overhead. Therefore, the model is suitable for deployment in resource-constrained IoT environments.

Table 7. Performance comparison with existing lightweight IDS models

Dataset	Ref.	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	No. Parameters	FLOPs
CICIDS2017	[10]	99.67	99.54	99.67	99.59	1654	610800
	[16]	99.84	99.84	99.84	99.84	14494	145297408
	Ours	**99.85**	**99.89**	**99.85**	**99.93**	13573	**26752**
NSL-KDD	[16]	97.55	97.50	97.55	97.49	14494	145297408
	[17]	98.66	95.22	85.68	89.03	4940	194580
	[18]	98.79	95.37	87.33	91.32	5070	198650
	Ours	**99.21**	**99.24**	**99.21**	**99.22**	13573	**26752**

5. Discussion

The proposed lightweight intrusion detection model detects various attack categories in IoT network data using a two-layer Multi-Layer Perceptron embedding backbone and a linear SoftMax classification head. The model has several advantages, including simplicity, efficiency, and excellent classification performance. The compact MLP design allows the model to learn discriminative representations of network traffic features while being lightweight. This design enables the model to achieve high detection performance with fewer parameters and lower computational overhead than many other deep learning algorithms. Experimental results on the CICIDS2017 and NSL-KDD datasets show that the proposed model achieves high classification accuracy while utilizing fewer FLOPs and MAC operations and having a modest model size. These aspects make the model suitable for use in resource-constrained IoT environments with limited memory and processing power. Furthermore, the comparatively low inference time enables more rapid identification of malicious traffic, which is critical for real-time network security monitoring.

Although it has several advantages, the proposed method has several limitations. The model performs poorly at detecting some attack types that exhibit traffic patterns similar to those of benign network behavior. This suggests that lightweight models continue to struggle to recognize subtle variations across traffic classes. Furthermore, the current design relies on a simplistic feature-embedding structure, limiting the model's ability to capture the complex traffic patterns in large-scale

 doi:10.20944/preprints202603.1915.v1

IoT networks. Future work will employ additional strategies to optimize detection capability while maintaining computational efficiency. Advanced feature selection, model pruning, quantization, and parameter optimization will be used to reduce the number of parameters and overall model size. These enhancements would make the model more viable for real-world IoT applications with limited resources.

6. Conclusions

To address security challenges in IoT networks, this work proposes a lightweight intrusion detection model based on a two-layer Multi-Layer Perceptron with an embedding backbone, followed by a linear SoftMax classification head for multi-class attack detection. The proposed approach learns compact, discriminative representations of network traffic attributes while remaining lightweight in architecture. The approach optimizes intrusion detection for resource-constrained IoT systems by minimizing the number of parameters and computational operations. The suggested method performs well on the CICIDS2017 and NSL-KDD datasets, achieving accuracies of 99.85% and 99.21%, respectively, while maintaining minimal computational complexity, parameter count, and model size.

Future studies will focus on improving the generalization capability of the proposed model across more varied and extensive network scenarios to increase its applicability in real-world IoT systems further. This includes testing the model on additional real-world datasets and strengthening it against complex, emerging cyber threats. Future studies will also investigate optimization techniques, such as feature selection, parameter tuning, model pruning, and quantization, to minimize the number of parameters and overall model size while maintaining exceptional detection performance.

Author Contributions: Conceptualization, Jisi Chandroth; Data curation, Jisi Chandroth; Funding acquisition, Jehad Ali; Investigation, Jehad Ali; Methodology, Jisi Chandroth; Project administration, Jehad Ali; Resources, Jehad Ali; Software, Jisi Chandroth; Supervision, Jehad Ali; Validation, Jehad Ali; Visualization, Jehad Ali; Writing – original draft, Jisi Chandroth; Writing – review & editing, Jehad Ali. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets analyzed in this study are publicly available from the following sources: (i) University of New Brunswick (UNB) CICIDS2017 dataset: https://www.unb.ca/cic/datasets/ids-2017.html; (ii) University of New Brunswick (UNB) NSL-KDD dataset: https://www.kaggle.com/datasets/hassan06/nslkdd.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Vishwakarma, A. K.; Chaurasia, S.; Kumar, K.; et al. Internet of Things Technology, Research, and Challenges: A Survey. *Multimed. Tools Appl.* **2025**, *84*, 8455–8490.
2. Secnora. BADBOX 2.0 Android Malware Infects Millions of Consumer Devices. July 18, 2025. Available online: https://secnora.com/blog/badbox-2-0-android-malware-infects-millions-of-consumer-devices/ (accessed on March 4, 2026).
3. Seals, T. Reaper Botnet Has Come for the Internet. October 21, 2017. Available online: https://www.infosecurity-magazine.com/news/reaper-botnet-has-come-for-the/ (accessed on March 4, 2026).
4. Rahman, M.; Al Shakil, S.; Mustakim, M. R. A Survey on Intrusion Detection System in IoT Networks. *Cyber Secur. Appl.* **2025**, *3*, 100082.
5. Hozouri, A.; Mirzaei, A.; Effatparvar, M. A Comprehensive Survey on Intrusion Detection Systems with Advances in Machine Learning, Deep Learning and Emerging Cybersecurity Challenges. *Discov. Artif. Intell.* **2025**, *5*, 314.
6. Xu, Z.; Wu, Y.; Wang, S.; Gao, J.; Qiu, T.; Wang, Z.; Wan, H.; Zhao, X. Deep Learning-Based Intrusion Detection Systems: A Survey. *arXiv* **2025**, arXiv:2504.07839.
7. Aldaej, A.; Ahanger, T. A.; Ullah, I. Deep Learning-Inspired IoT-IDS Mechanism for Edge Computing Environments. *Sensors* **2023**, *23*, 9869.

 doi:10.20944/preprints202603.1915.v1

8. Rawat, M.; Singal, G. Surveying Technology Fusion in IoT Networks for IDS: Exploring Datasets, Tools, Challenges, and Research Prospects. *ACM Trans. Intell. Syst. Technol.* **2025**, *16*, 107.
9. Ali, J.; Song, H. H.; Sharma, V.; Al-Khasawneh, M. A. DDoS Intrusions Detection in Low Power SD-IoT Devices Leveraging Effective Machine Learning. *IEEE Transactions on Consumer Electronics* **2024**, *71*, 343–351.
10. Wang, Z.; Chen, H.; Yang, S.; Luo, X.; Li, D.; Wang, J. A Lightweight Intrusion Detection Method for IoT Based on Deep Learning and Dynamic Quantization. *PeerJ Comput. Sci.* **2023**, *9*, e1569.
11. Omarov, B.; Auelbekov, O.; Suliman, A.; Zhaxanova, A. CNN-BiLSTM Hybrid Model for Network Anomaly Detection in Internet of Things. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 349.
12. Fatima, M.; Rehman, O.; Rahman, I. M. H.; Ajmal, A.; Park, S. J. Towards Ensemble Feature Selection for Lightweight Intrusion Detection in Resource-Constrained IoT Devices. *Future Internet* **2024**, *16*, 368.
13. Wang, Z.; Zhou, R.; Yang, S.; He, D.; Chan, S. A Novel Lightweight IoT Intrusion Detection Model Based on Self-Knowledge Distillation. *IEEE Internet Things J.* **2025**, *12*, 16912–16930.
14. Govindarajan, V.; Ahmed, F.; Faheem, Z. B.; Bilal, M.; Ayadi, M.; Ali, J. Aegis-5: A Hybrid Ensemble Framework for Intrusion Detection in Industry 5.0 Driven Smart Manufacturing Environment. *ACM Transactions on Autonomous and Adaptive Systems* **2026**.
15. Benaddi, H.; Jouhari, M.; Elharrouss, O. A Lightweight Hybrid Approach for Intrusion Detection Systems Using a Chi-Square Feature Selection Approach in IoT. *Internet of Things* **2025**, *32*, 101624.
16. Li, Z.; Yao, W. A Two-Stage Lightweight Approach for Intrusion Detection in Internet of Things. *Expert Systems with Applications* **2024**, *257*, 124965.
17. Yang, S.; Zheng, X.; Xu, Z.; Wang, X. A Lightweight Approach for Network Intrusion Detection Based on Self-Knowledge Distillation. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Rome, Italy, May 2023; pp. 3000–3005.
18. Feng, S.; Ma, S.; Ma, M. A Lightweight Network Intrusion Detection Method Based on Protocol-Aware Dynamic Inverted Residuals and a Sliding-Window Multi-Batch Self-Knowledge Distillation Strategy. In *Proceedings of the Guangdong-Hong Kong-Macao Greater Bay Area Education Digitalization and Computer Science International Conference (EDCS)*, Guangdong, China, 2025; pp. 833–838.
19. Khan, A.; Hussain, M. A.; Anwer, F. A Hybrid Lightweight Deep Learning-Based Intrusion Detection Approach in IoT Utilizing Feature Selection and Explainable Artificial Intelligence. *IEEE Access* **2025**, *13*, 192451–192466.
20. Yang, X.; Tong, F.; Jiang, F.; Cheng, G. A Lightweight and Dynamic Open-Set Intrusion Detection for Industrial Internet of Things. *IEEE Transactions on Information Forensics and Security* **2025**, *20*, 2930–2943.
21. Ma, W.; Wang, X.; Dong, J.; Hu, M.; Zhou, Q. A Lightweight Method for Botnet Detection in Internet of Things Environment. *IEEE Transactions on Network Science and Engineering* **2025**, *12*, 2458–2472.
22. Zhang, X.; Wang, Y.; Han, G.; Gui, G. Advanced Few-Shot Network Intrusion Detection Method Using Lightweight Transfer Learning. *IEEE Internet of Things Journal* **2025**, *12*, 48678–48688.
23. Fard, E.; Soltani, M.; Jahangir, A. H.; et al. LightIDS: A Lightweight Neural Network-Based Intrusion Detection System. *Journal of Supercomputing* **2026**, *82*, 18.
24. Wang, L.-H.; Dai, Q.; Du, T.; Chen, L.-F. Lightweight Intrusion Detection Model Based on CNN and Knowledge Distillation. *Applied Soft Computing* **2024**, *165*, 112118.
25. Iliyasu, A. S.; Siddiqui, A. J.; Song, H.; Abdu, F. J. PNet-IDS: A Lightweight and Generalizable Convolutional Neural Network for Intrusion Detection in Internet of Things. *IEEE Access* **2025**, *13*, 102624–102639.
26. Pasquini, A.; Vasa, R.; Logothetis, I.; Habibi Gharakheili, H.; Chambers, A.; Tran, M. Robust and Lightweight Modeling of IoT Network Behaviors From Raw Traffic Packets. *IEEE Transactions on Machine Learning in Communications and Networking* **2025**, *3*, 98–116.
27. Alhassan, A. M. Self-Adaptive Lightweight Attention Module-Based BiLSTM Model for Effective Intrusion Detection. *Arabian Journal for Science and Engineering* **2025**, *50*, 11513–11538.
28. Sharafaldin, I.; Lashkari, A. H.; Ghorbani, A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Portugal, 22–24 January 2018; pp. 108–116.
29. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A. A. A Detailed Analysis of the KDD CUP 99 Data Set. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Ottawa, ON, Canada, July 2009; pp. 1–6.

 doi:10.20944/preprints202603.1915.v1

disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.