Lightweight Deep Learning Model for DDoS Detection in IoT Environments

Mina Habibollahi Rohith Perumandla Swetha Manupati Mary Vanco
DePaul University
mhabibol@depaul.edu, rohith.perumandla.12@gmail.com, smanupat@gmail.com, maryvanco@gmail.com

Abstract

The rapid proliferation of IoT devices has intensified the risk of Distributed Denial-of-Service (DDoS) attacks, underscoring the need for detection systems that are both accurate and lightweight enough for real-time deployment on resource-constrained hardware. This paper presents a two-stage DDoS detection framework designed specifically for IoT environments. In the first stage, we establish a baseline using Principal Component Analysis (PCA) for dimensionality reduction combined with a Multilayer Perceptron (MLP) classifier. In the second stage, a deep Autoencoder is used to extract compact feature representations, followed by lightweight classifiers including Logistic Regression, MLP, and a compact Convolutional Neural Network (CNN). The Autoencoder effectively reduces the feature set to five dimensions, significantly improving inference speed. Experimental results demonstrate that all Autoencoder-based models outperform the PCA-based baseline in efficiency. Notably, our lightweight MLP with five Autoencoder features achieves the best trade-off, with 98.88% accuracy and an inference time of just 6.25 µs per sample. These results highlight the potential of our method for real-time, resource-efficient DDoS detection in IoT systems

1. Introduction

Distributed Denial-of-Service (DDoS) attacks pose a serious and growing threat to the availability and stability of networked systems, particularly within the rapidly expanding Internet of Things (IoT) ecosystem. These attacks flood target devices or networks with illegitimate traffic, disrupting essential services and potentially causing severe operational and financial damage. The threat is especially critical for IoT devices, which are often limited in processing power, memory, and built-in security. As IoT technologies are increasingly integrated into vital infrastructure, such as healthcare, transportation, and smart cities, the need for effective and efficient DDoS detection mechanisms becomes paramount.

Deep neural networks (DNNs) have shown impressive performance in detecting DDoS attacks, but their high computational demands make them unsuitable for deployment on resource-constrained IoT devices. This creates a crucial gap: the need for lightweight models that maintain strong detection performance while being efficient enough for real-time use in low-power environments. Furthermore, many existing studies emphasize detection accuracy but often overlook essential deployment metrics such as training and testing time, model size, and number of features, all of which directly impact the model's suitability for IoT devices.

To address these challenges, we propose a novel and efficient approach to DDoS detection tailored for IoT networks. Our methodology includes two stages. First, we establish a baseline using Principal Component Analysis (PCA) for feature reduction and a Multilayer Perceptron (MLP) as the classifier. In the second stage, we develop an optimized, lightweight deep learning system. This system employs a deep Autoencoder for extracting low-dimensional feature representations, followed by lightweight classifiers including a compact Convolutional Neural Network (CNN), Logistic Regression (LR), and MLP. We evaluate the performance trade-offs between accuracy, testing/training time, and model simplicity.

Our main contributions are as follows:

1-Lightweight Feature Extraction: We reduce the feature set to only 5 dimensions using an Autoencoder, which is significantly lower than most related works, demonstrating the compactness and efficiency of our model.

2-Comprehensive Evaluation: Unlike many prior studies, we measure both training and testing inference time to assess the true deployability of our models in real-world IoT environments.

3-Model Comparisons: We provide a detailed comparison between a traditional baseline model (PCA + MLP) and our proposed architecture, analyzing trade-offs in accuracy, speed, and resource consumption.

Our results demonstrate that it is possible to maintain high detection accuracy for DDoS attacks while significantly reducing the computational cost, making our approach highly suitable for real-time applications in IoT en-

vironments.

2. Related Works

Detecting Distributed Denial-of-Service (DDoS) attacks in Internet of Things (IoT) environments has become a critical area of research due to the increasing prevalence of cyber threats targeting resource-constrained devices. Numerous studies have leveraged deep learning (DL) and machine learning (ML) techniques to improve detection accuracy while attempting to balance computational efficiency, which is essential for practical deployment on IoT devices. In this section, we review recent relevant works that focus on lightweight and effective models for DDoS detection in IoT networks, highlighting their strengths and limitations with respect to feature complexity, training and inference time, and suitability for resource-limited environments.

This work by [1] introduces CBCO-ERNN, an optimized Elman Recurrent Neural Network designed for detecting Distributed Denial-of-Service (DDoS) attacks in IoT environments. The model utilizes chaotic bacterial colony optimization (CBCO) to fine-tune its parameters, achieving high accuracy, precision, and F-score across four benchmark datasets (BoT-IoT, CIC-IDS2017, CIC-DDoS2019, IoTID20). However, a key limitation is its reliance on 80 input features, which significantly increases computational complexity and hinders its suitability for lightweight IoT devices. Furthermore, the study lacks evaluation of prediction latency or inference time, crucial metrics for real-world deployment on resource-constrained IoT systems.

This study[2], presents a lightweight CNN-based model for detecting DDoS attacks, optimized for low processing overhead and fast detection, making it suitable for real-world applications. The model utilizes automatic feature extraction, max pooling, flattening, and weight sharing to reduce complexity and memory usage. It also supports dynamic parameter tuning for resource-constrained environments and achieves high accuracy, processing over 55,000 samples per second. However, the authors rely on grid search for hyperparameter tuning, which is computationally expensive and increases training time. In contrast, our work uses Optuna with early stopping and pruning strategies to significantly reduce the training time of the autoencoder while maintaining competitive performance.

This study [3] focuses on enhancing IoT intrusion detection systems by leveraging advanced deep learning models. The researchers proposed an ensemble framework combining CNNs, LSTMs, and GRUs, with a voting mechanism to capture complex hierarchical patterns in network traffic data. Evaluated on the NSL-KDD dataset, their hybrid models, CNN-LSTM and CNN-GRU, achieved outstanding accuracy rates of 99.7% and 99.6%, with corresponding F1 scores of 0.998 and 0.997, outperforming standalone CNN and LSTM models. For hyperparameter optimization, the study employed the Adaptive Particle Swarm Optimization (APSO) algorithm. However, critical implementation details such as training time, testing time, number of features used were not reported.

This study[4], proposes the Associated Random Neural Network (ARNN) to enhance IoT network security by collectively detecting compromised devices through fully recurrent connections and paired neurons per node. Trained on real-world botnet attack data, ARNN achieves outstanding accuracy (up to 100%), fast detection (3.5 ms per node), and superior performance over models like CNN, LSTM, and CDIS-DRNN. It reduces input complexity by aggregating six traffic metrics into a single scalar per neuron and supports both offline and online learning. However, its key drawback lies in its high training complexity due to the need for inverting large matrices which makes it significantly slower to train compared to conventional models. In contrast, our training time is expected to be substantially less than this model's, enabling faster development and deployment.

This paper[5], proposes a novel integrated model based on a deep autoencoder (AE) for both anomaly detection and feature extraction in IoT network traffic. The AE is first trained on normal traffic to detect anomalies and subsequently used to extract low-dimensional features from anomalous data without requiring a separate feature extraction training phase, unlike PCA or LDA. These extracted features are then fed into various machine learning and deep learning classifiers, including Decision Tree, Random Forest, DNN, CNN, and CNN-LSTM, for multi-class attack classification.Among the evaluated models, the Autoencoder with 8 extracted features (AE-8) achieved the best performance, with an accuracy of 90%, recall of 92%, F1-score of 90%, and precision of 87%. The AE-8 model required a training time of 8.8 seconds per sample and achieved a testing time of approximately 0.03 seconds per sample. However, the AE-8 model's accuracy remains relatively moderate considering it uses 8 features. In contrast, our approach extracts only 5 features, resulting in a more lightweight model that is better suited for deployment in resource-constrained IoT environments, while maintaining competitive or superior performance.

3. Preliminary / Background

3.1. Distributed Denial-of-Service (DDoS) Attacks in IoT

The Internet of Things (IoT) consists of interconnected devices that communicate and share data autonomously. While IoT offers enormous benefits across domains such as healthcare, transportation, smart cities, and industrial control, it also introduces serious security vulnerabilities. One of the most common and severe threats in IoT networks is

Distributed Denial-of-Service (DDoS) attacks, where large volumes of malicious traffic are generated to overwhelm and disable target devices or entire network infrastructures.

IoT devices are particularly susceptible to DDoS attacks due to their limited computational power, memory, and often insufficient security protocols. These resource constraints make it challenging to deploy traditional heavyweight detection models directly on IoT devices, motivating the need for lightweight and efficient detection mechanisms.

3.2. Machine Learning for DDoS Detection

Machine learning (ML) and deep learning (DL) have become prominent tools for detecting DDoS attacks by learning patterns from network traffic data. Traditional ML models, such as Logistic Regression, Decision Trees, and Support Vector Machines, can effectively classify attack traffic if provided with relevant features. Deep learning models, such as Convolutional Neural Networks (CNNs) and Multilayer Perceptrons (MLPs), offer superior performance by automatically learning complex patterns from raw or processed features.

However, many of these models are computationally intensive and may not be suitable for real-time deployment on IoT devices. As a result, designing lightweight yet accurate models remains an active area of research for IoT security.

3.3. Dimensionality Reduction

High-dimensional network traffic data often contain redundant or irrelevant features, which may degrade model performance and increase computational cost. Dimensionality reduction techniques are widely used to simplify the input feature space while preserving essential information:

- **Principal Component Analysis (PCA):** A statistical technique that transforms the original features into a smaller set of orthogonal components, capturing the maximum variance in the data.
- **Autoencoders:** Neural network-based unsupervised models that compress input data into a low-dimensional latent representation and reconstruct the input, preserving important patterns while reducing dimensionality.

In this work, we explore both PCA and Autoencoder-based feature extraction approaches to enable lightweight DDoS detection models.

3.4. Problem Formulation

Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ represents the network traffic feature vector and $y_i \in \{0, 1\}$ represents the binary label indicating benign traffic (0) or DDoS attack (1), the goal is to learn a function $f : \mathbb{R}^d \rightarrow \{0, 1\}$ that accurately classifies incoming traffic.

This classification problem can be approached by first reducing the dimensionality of x_i to a lower-dimensional feature space $z_i \in \mathbb{R}^k$ (where $k \ll d$) using PCA or Autoencoders. The transformed features z_i are then used to train lightweight classifiers such as MLP, Logistic Regression, and CNNs to perform binary classification.

3.5. Preliminary Assumptions

In developing our methodology, we make the following assumptions:

- The dataset is representative of the actual population of network traffic, capturing realistic patterns of both benign and DDoS attack behaviors commonly encountered in IoT environments.
- Proper preprocessing steps (duplicate removal, missing value handling, feature cleaning, class balancing, and standardization) have been applied to ensure data quality and consistency.
- Dimensionality reduction techniques such as PCA and Autoencoder-based feature extraction are capable of preserving the most critical information from the original feature space while discarding irrelevant or redundant features.
- Reducing the feature dimensionality leads to simpler, more compact models that are better suited for deployment on resource-constrained IoT devices, as it lowers memory consumption, computational complexity, and inference latency.
- The extracted lower-dimensional feature representations retain sufficient discriminatory power to allow the downstream classifiers to accurately differentiate between benign and DDoS traffic.

These assumptions form the foundation for our proposed lightweight detection framework presented in the subsequent sections.

4. Methodology

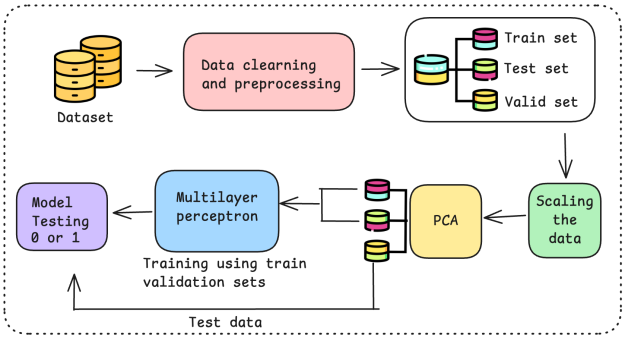

Figure 1. Baseline model architecture: The original dataset undergoes preprocessing, splitting, scaling, and dimensionality reduction using PCA. The processed data is then used to train and evaluate a Multilayer Perceptron (MLP) model, which serves as our benchmark for comparison.

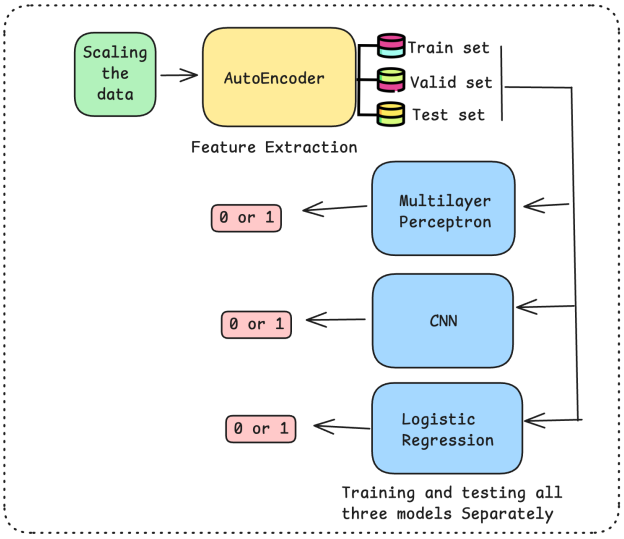

Figure 2. Proposed optimized architecture: After data scaling, an autoencoder extracts compressed latent features, which are then used to train and evaluate three individual models: MLP, CNN, and Logistic Regression. Their predictions are combined using majority voting to produce the final DDoS attack detection output. This design supports lightweight deployment in IoT environments.

Our proposed approach consists of a two-stage methodology designed to balance detection accuracy with computational efficiency for real-time DDoS detection in IoT environments. In the first stage, we establish a baseline model using dimensionality reduction and a traditional neural network classifier illustrated in Figure 1. In the second stage, we build an optimized lightweight models leveraging deep feature extraction and multiple lightweight classifiers shown in Figure 2.

4.1. Dataset Source, Preprocessing, and Cleaning

For this study, we utilized a publicly available DDoS attack dataset collected from Kaggle, consisting of 7,948,748 samples and 84 features. The features capture detailed network traffic statistics such as packet sizes, flow duration, inter-arrival times, and protocol flags, along with the corresponding class labels for DDoS and benign flows.

Preprocessing Steps:

A comprehensive preprocessing pipeline was applied to clean and prepare the dataset for model training:

- **Duplicate Removal:** All duplicate records were removed to eliminate redundant data points that could bias the training process.
- **Data Type Optimization:** Feature data types were converted from 64-bit to 32-bit precision to reduce memory usage and improve computational efficiency.
- **Missing Value Handling:** Any missing values were handled and cleaned to ensure no invalid or incomplete records were passed into the model.
- **Feature Removal:** Non-informative or identifier features were dropped, including `Flow ID`, `Src IP`, `Dst IP`, `Timestamp`, and `Label`. These features do not provide meaningful information for learning attack patterns.
- **Class Balancing:** The original dataset was highly imbalanced, containing approximately 92.7% benign samples and only 7.3% DDoS attack samples. To avoid model bias, we performed random undersampling on the benign class to balance both classes. After balancing, both classes contained 576,191 samples each.
- **Dataset Splitting:** The balanced dataset was divided into training, validation, and test sets using a 80%-10%-10% split ratio. Stratified sampling was applied to preserve the class distributions across splits.
- **Standardization:** StandardScaler was applied to normalize all features to zero mean and unit variance, ensuring that features with different scales did not dominate the learning process. Importantly, the scaler was fitted only on the training set and then applied to validation and test sets to avoid data leakage.

4.2. Feature Extraction with PCA:

Given the high dimensionality of the original dataset (84 features), we employed Principal Component Analysis (PCA) to reduce the feature space while retaining maximum variance. PCA was fitted on the scaled training, test and validation set separately to prevent data leakage. A cumulative explained variance threshold of 95% was used, resulting in 20 principal components being retained for modeling. The PCA scree plot illustrating the explained variance is shown in Figure 3. This dimensionality reduction helps reduce model complexity, speed up training, and enable deployment of lightweight models on resource-constrained IoT devices.

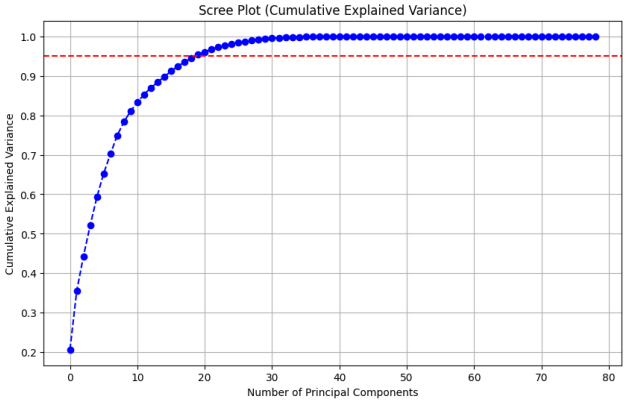

Figure 3. Scree plot showing cumulative explained variance from PCA. 20 principal components were selected to retain 95% of the total variance.

4.3. Baseline Model: Multilayer Perceptron (MLP) with PCA Features

After feature extraction using PCA, we trained a Multilayer Perceptron (MLP) model as our baseline classifier. The PCA-reduced dataset with 20 principal components served as the input feature space for this model.

Model Architecture: The MLP architecture was designed to be both simple and computationally efficient to reflect the lightweight deployment requirements of IoT environments. The architecture consisted of:

- **Input layer:** 20 PCA components.
- **Hidden Layer 1:** 64 neurons, followed by Batch Normalization, ReLU activation, and Dropout with rate 0.3.
- **Hidden Layer 2:** 32 neurons, followed by Batch Normalization, ReLU activation, and Dropout with rate 0.3.
- **Output Layer:** 1 neuron with Sigmoid activation for binary classification.

This model design was selected to balance model capacity while preventing overfitting through multiple regularization techniques. Batch Normalization was included to stabilize learning and improve convergence speed. Dropout layers were used to prevent co-adaptation of neurons and improve generalization.

Loss Function and Optimization: We used Binary Cross Entropy (BCE) loss as the objective function since this is a binary classification task. The model was trained using the AdamW optimizer, which combines adaptive learning rates with weight decay regularization to improve both convergence and generalization. The optimizer hyperparameters were: Learning Rate: 0.001, Weight Decay: 1×10^{-4}

Early Stopping: To prevent overfitting, early stopping was applied by monitoring the validation loss at each epoch. If the validation loss did not improve for 5 consecutive epochs (patience = 5), training was stopped, and the model with the lowest validation loss was saved as the final model.

Training Parameters:

- Batch Size: 64
- Maximum Epochs: 20
- Patience: 5 (for early stopping)

Hyperparameter Selection Rationale: While hyperparameter search methods (e.g., GridSearchCV or Optuna) were considered, the chosen default configuration provided strong performance without requiring extensive search, as discussed in the Results section.

The hyperparameters were selected based on prior research, widely accepted deep learning practices for tabular data, and empirical tuning for small structured datasets. The hidden layer sizes (64 and 32 neurons) were chosen to balance model capacity and overfitting risk for the 20 PCA components. ReLU activations were selected for their simplicity and efficient gradient flow properties. Batch Normalization was applied after each hidden layer to stabilize training and reduce internal covariate shift. Dropout regularization (30%) was incorporated to prevent co-adaptation of neurons. The AdamW optimizer was selected for its ability to handle noisy gradients, with a learning rate of 0.001 and light weight decay of 1×10^{-4} to ensure controlled regularization. The batch size (64) was chosen to balance computational efficiency and stability, while early stopping with a patience of 5 ensured optimal convergence without overfitting.

Evaluation Metrics: After training, the best model was evaluated on the test set using multiple classification metrics to comprehensively assess its performance, including Accuracy, Precision, Recall, F1-Score, AUC-ROC, Confusion Matrix, and ROC Curve. In addition to these metrics, inference time was also measured to assess the model's suitability for real-time deployment. These same evaluation metrics were consistently applied to all proposed models in order to ensure a comprehensive comparison.

This baseline model serves as a performance reference for comparison with our proposed optimized models trained on features extracted using Autoencoder.

5. Proposed Lightweight Architecture

To address the computational limitations of IoT devices while maintaining strong DDoS detection performance, we propose a lightweight two-stage architecture, illustrated in Figure 2. In this design, an Autoencoder is first used to extract a compact and meaningful latent representation from the original high-dimensional feature space. These extracted features are then used as input to three separate classifiers: a Multilayer Perceptron (MLP), a lightweight Convolutional Neural Network (CNN), and a Logistic Regression model. Finally, a majority voting mechanism combines the outputs of these models to generate the final prediction. This design enhances both accuracy and robustness while ensuring suitability for real-time deployment in resource-constrained IoT environments.

In addition to evaluating classification performance, we also measure the inference time of each individual model to assess the deployability and real-time suitability of the proposed architecture in resource-constrained IoT environments.

5.1. Autoencoder for Feature Extraction

The goal of this component is to extract a compact, informative representation of the input data that can serve as input for downstream classifiers. To achieve this, we employ an autoencoder, a neural network architecture designed for unsupervised feature learning and dimensionality reduction.

5.1.1 Autoencoder Architecture

An autoencoder consists of two main components:

- **Encoder:** Compresses the input features into a lower-dimensional latent representation.
- **Decoder:** Reconstructs the original input from the latent vector.

The model is trained by minimizing the Mean Squared Error (MSE) between the original input and its reconstruction. Once trained, the encoder part is retained to transform raw input data into its compressed, latent form. This latent vector preserves the most meaningful patterns from the input and serves as the extracted feature representation.

5.1.2 Hyperparameter Optimization with Optuna

To ensure that the autoencoder is well-optimized and generalizes well, we employed *Optuna*, a state-of-the-art hyperparameter optimization framework. The following parameters were included in the search space:

- `hidden_dim`: number of neurons in the hidden layer
- `latent_dim`: bottleneck size or number of extracted features (searched in the range 2 to 7)
- `learning rate`: optimizer step size
- `dropout_rate`: regularization to prevent overfitting
- `batch_size`: size of each mini-batch during training
- `activation_fn`: type of activation function (`ReLU` or `LeakyReLU`)

Each configuration was evaluated over 5 training epochs using a training-validation split, and the reconstruction error was used as the objective function to minimize. A Median Pruner was applied to stop poorly performing trials early and improve search efficiency.

5.1.3 Latent Dimensionality Selection

During hyperparameter tuning, we specifically explored the range of latent dimensionality from 2 to 7 to determine the optimal number of features for the bottleneck layer. This range allowed us to balance model compactness and reconstruction quality.

As a result, Optuna selected a latent dimension of 5, indicating that a 5-dimensional latent space yielded the best trade-off between compactness and information preservation. These five features were chosen as the final feature set for downstream classification tasks.

5.1.4 Final Model Training and Feature Extraction

Using the optimal configuration identified by Optuna, we retrained the autoencoder on the full training dataset for 50 epochs to ensure convergence and stable learning. The encoder portion of the trained model was then used to extract latent features using the `extract_features()` method.

The final output of this process is a feature matrix of shape `(n_samples, 5)`, where each row represents a compressed and meaningful representation of the original input. This representation significantly reduces computational overhead while preserving essential discriminative information, making it ideal for lightweight and accurate DDoS detection in IoT environments.

5.2. Multilayer Perceptron (MLP) Classifier

We trained a Multilayer Perceptron (MLP) model on features extracted using autoencode. This model operates on the 5-dimensional feature representations extracted by the Autoencoder.

The same MLP architecture and hyperparameter configuration described in Section 4.3 was adopted here to ensure consistency across experimental comparisons. Specifically, only the input layer size was adjusted to match the reduced dimensionality of the Autoencoder features. The rest of the architecture including the number of neurons in hidden layers, activation functions, batch normalization, dropout rate, optimizer choice, learning rate, batch size, and early stopping strategy remained identical to the baseline model.

By reusing the same configuration, we ensured that any differences in performance between the baseline model and these optimized models could be attributed primarily to the feature extraction method (PCA vs. Autoencoder), rather than changes in model complexity or hyperparameter tuning. The evaluation of this model was conducted using the same metrics and inference time analysis as outlined previously. The results obtained for this model are presented and discussed in the Results section.

5.3. Convolutional Neural Network (CNN) Classifier

The lightweight CNN model was designed to detect patterns in the 5 dimensional features extracted from the Autoencoder.

Model Architecture The model uses two 1D convolutional layers to slide small filters across the input, identifying local dependencies between features that may indicate DDoS attacks. Each convolutional layer is followed by a ReLU activation and a max pooling layer to reduce dimensionality and highlight the most informative patterns, making the model faster and less prone to overfitting. After flattening the output, the data passes through a small fully connected layer with ReLU activation and a dropout layer (30%

rate) for regularization. Finally, a sigmoid-activated output layer produces a probability for binary classification. This lightweight structure balances detection accuracy with computational efficiency, making it suitable for real-time use in resource-constrained environments.

Loss Function and Optimizer Binary Cross Entropy was used as the loss function and Adam optimizer was used to train the CNN. Binary Cross Entropy Loss is well-suited for binary classification tasks, where the model outputs a probability between 0 and 1 indicating whether a given network input represents an attack. The Adam optimizer is chosen because it adapts the learning rate for each parameter individually, leading to faster and more stable convergence, which is important when training on large, complex datasets. This combination of loss function and optimizer balances efficiency and performance.

Hyperparameter Selection Rationale: While automated hyperparameter tuning techniques were considered, the final hyperparameters for each model were selected based on research and best practices for similar deep learning tasks. A learning rate of 0.001 was chosen to ensure stable convergence. A batch size of 64 was used to balance memory efficiency and training speed. Dropout rate of 30% were applied to reduce overfitting, and early stopping with a patience of 3 epochs was implemented to prevent unnecessary training once validation loss plateaued. Although a full hyperparameter grid search was not conducted, these carefully selected values allowed the models to achieve high accuracy and robust performance as indicated in the results.

5.4. Logistic Regression Classifier

As part of our proposed lightweight architecture, in addition to the two deep learning models (MLP and CNN), we include a Logistic Regression classifier to provide a highly efficient, interpretable, and computationally inexpensive baseline. The Logistic Regression model was trained on the 5-dimensional features extracted from the Autoencoder.

To ensure optimal performance, we first combined the training and validation datasets and used 5-fold cross-validation during hyperparameter tuning. Since Logistic Regression models are typically fast to train, we employed GridSearchCV to systematically explore a range of hyperparameter combinations. The parameter grid was defined as follows:

- `C`: [0.001, 0.01, 0.1, 1, 10, 100]
- `penalty`: ['l1', 'l2', None]
- `solver`: ['lbfgs', 'liblinear']
- `max_iter`: [500, 1000]

The hyperparameter `C` controls the inverse of the regularization strength. Smaller values of `C` apply stronger regularization, while larger values allow the model to fit the data more flexibly. Since we are working with only 5 highly informative features extracted by the Autoencoder, we tested a wide range of `C` values from 0.001 to 100 to allow the model flexibility while also considering regularization to prevent overfitting. The `penalty` parameter was used to explore different regularization types (L1, L2, or no regularization), while `solver` specifies the optimization algorithm compatible with each penalty type. The `max_iter` parameter was set to 500 and 1000 to ensure convergence.

After performing GridSearchCV, the best hyperparameters selected were:

- `max_iter`: 1000
- `solver`: lbfgs
- `penalty`: None

The selection of `penalty=None` indicates that no regularization was necessary for this model. This can be explained by the fact that the Autoencoder has already performed feature extraction and dimensionality reduction, leaving only 5 highly informative and uncorrelated features. As a result, Logistic Regression was able to fit the data directly without overfitting, eliminating the need for regularization.

Using these optimized hyperparameters, the Logistic Regression model was trained and evaluated on the test set. The final performance results for this model are reported in the Results section.

6. Numerical Experiments

Table 1. Performance metrics comparison across models.

Model	Accuracy	Precision	Recall	F1	AUC
Baseline MLP (PCA)	99.85%	99.87%	99.83%	99.85%	99.99%
Logistic Regression	90.81%	84.80%	99.46%	91.54%	90.71%
MLP (Autoencoder)	98.88%	97.83%	99.98%	98.89%	99.98%
CNN (Autoencoder)	99.85%	99.82%	99.87%	99.85%	99.93%

Table 2. Inference time comparison across models (in seconds).

Model	Average Inference Time Per Sample (μs)
Baseline MLP (PCA)	7.4
Logistic Regression	0.022
MLP (Autoencoder)	6.3
CNN (Autoencoder)	12.8

6.1. Performance and Inference Time Analysis

Table 1 summarizes the classification performance across models, while Table 2 compares inference times. The Baseline MLP model trained on PCA-reduced features (20 principal components) achieves excellent performance with 99.85

In terms of inference efficiency (Table 2), Logistic Regression demonstrates the fastest inference time of only 0.022 μs per sample due to its linear structure and minimal parameter size. The MLP (Autoencoder) is the fastest among deep learning models at 6.3 μs, followed by the PCA-based MLP at 7.4 μs, while CNN (Autoencoder) takes slightly longer at 12.8 μs due to its convolutional operations.

Figure 4 visually compares the models across model size, inference speed, and F1-score. As seen, the Autoencoder-based models strike an excellent trade-off: significant feature compression (from 84 raw features to just 5), while achieving both high predictive performance and extremely low latency — making them well-suited for IoT real-time DDoS detection. The PCA-based baseline performs slightly better in F1-score, but requires more input dimensions (20) and larger model size, while CNN (Autoencoder) achieves superior robustness but at slightly higher computational cost.

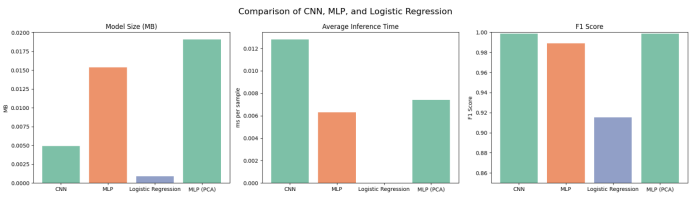

Figure 4. Comparing the F1-Score, Avg Inference time and Model size

7. Conclusion

In this study, we introduced a lightweight and efficient DDoS detection framework specifically designed for IoT environments, where computational resources are limited and real-time processing is essential. Our two-stage approach integrates dimensionality reduction and classification, balancing detection accuracy, inference speed, and model complexity. By leveraging a deep Autoencoder to compress the feature space from 84 original features to only 5 dimensions, we significantly improved model efficiency without sacrificing performance.

Among the evaluated models, the Autoencoder-based MLP achieved the best trade-off, reaching 98.88% accuracy with an average inference time of just 6.3 microseconds per sample. This performance not only outperformed the PCA-based MLP baseline but also demonstrated competitive results compared to CNN-based models while maintaining a much smaller model size. Although logistic regression achieved the fastest inference time, its lower classification performance with AUC 90% highlights the limitations of linear models for complex network traffic patterns, whereas deep learning models are better suited to capture intricate feature interactions present in DDoS attacks.

While these results are promising, the current evaluation was limited to a single dataset and controlled simulation environment. Future work will focus on validating the proposed framework on additional public and proprietary datasets, as well as deploying and testing the model directly on real-world IoT hardware to further assess its generalizability and practical deployment viability.

References

[1] Khaled A. Alaghbari, Heng-Siong Lim, Mohamad Hanif Md Saad, and Yik Seng Yong. Deep autoencoder-based integrated model for anomaly detection and efficient feature extraction in iot networks. *IoT*, 4(3):345–365, 2023. 8

[2] Roberto Doriguzzi-Corin, Stuart Millar, Sandra Scott-Hayward, Jesus Martinez-del Rincon, and Domenico Siracusa. LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Transactions on Network and Service Management*, 17(2):876–889, 2020. 8

[3] Erol Gelenbe and Mert Nakip. Iot network cybersecurity assessment with the associated random neural network. *IEEE Access*, 11:85501–85512, 2023. 8

[4] MI Thariq Hussan, G Vinoda Reddy, PT Anitha, A Kanagaraj, and Pannangi Naresh. Ddos attack detection in iot environment using optimized elman recurrent neural networks based on chaotic bacterial colony optimization. *Cluster Computing*, 27(4):4469–4490, 2024. 8

[5] Ammar Odeh and Anas Abu Taleb. Ensemble-based deep learning models for enhancing iot intrusion detection. *Applied Sciences*, 13(21):11985, 2023. 8