# Lightweight Anomaly Detection for IoT: Evaluating Machine Learning and Deep Learning Models on CICIDS2017

Emanuele Iacobelli[1 [0009−0003−1379−9106]], Valerio Ponzi[1,2 [0009−0000−2910−0273]], Adriano Puglisi[1 [0009−0007−6307−7194]], Oleksandr Kuznetsov[4 [0000−0003−2331−6326]], Katarzyna Nieszporek[3 [0000−0002−9424−2646]], Cristian Randieri[1,4 [0000−0001−5300−3561]], and Christian Napoli[1,2,3 [0000−0002−3336−5853]]

[1] Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy
[2] Institute for Systems Analysis and Computer Science, Italian National Research Council (CNR), 00185 Rome, Italy
[3] Department of Intelligent Computer Systems, Czestochowa University of Technology, 42-200 Czestochowa, Poland
[4] eCampus University, Via Isimbardi 10, 22060 Novedrate (CO), Italy
cnapoli@diag.uniroma1.it

**Abstract.** Given the increasing rate of cyber attacks, specifically Denial of Service (DoS) attacks, there is a growing need for fast and efficient Intrusion Detection Systems (IDS). In this work, we studied the implementation of real-time IDS within resource constrained environments like Internet of Things (IoT) networks. We studied and tested a wide range of Machine Learning and Deep Learning models applied to the CICIDS2017 dataset, a commonly used benchmarking tool for network intrusion detection. We compared the results of models such as Logistic Regression, Random Forest, XGBoost, K-Nearest Neighbors, Support Vector Machines, Single-layer Perceptron (SLP), Multi-layer Perceptron (MLP), Deep Convolutional Neural Network (DCNN), ResNet, and DenseNet. We focused our investigation on performance metrics such as accuracy, precision, recall, F1-score, and inference time, trying to find the model with the best trade-off between detection capability and computation overhead considering the constrained resources of IoT devices. The results highlight that real-time security of IoT infrastructures with minimal resource consumption is possible with simple models such as XGBoost, SLP, or MLP.

**Keywords:** Artificial Intelligence · Machine Learning · Deep Learning · DoS Attacks · Internet of Things

## 1 Introduction

The term Internet of Things (IoT) was introduced in 1999 by Kevin Ashton to describe a network of physical devices connected to internet through sensors.

Nowadays, this concept has evolved in a lot of areas of application, such as smart home systems, healthcare, transport, manufacturing processes, and more, generating greater efficiency and automation. However, being intrinsically interconnected through internet, expose these devices to possible cyber attacks, making cyber security a necessity. Common cyber attacks range from data theft to the more sophisticated Distributed Denial-of-Service (DDoS) attacks [1], creating problems for digital infrastructure, financial security, and individual privacy. Among the many cyber threats, intrusion attacks, in which unauthorized access to a system or network occurs, are of particular concern. Signature-based and anomaly-based intrusions can be characterized as general categories of intrusion attacks. Signature-based methods consist of established attack patterns, while anomaly-based detection involves looking for variations from normal patterns and is therefore a promising way to discover unknown threats.

Traditional security measures, typically reliant on cloud-based analysis, introduce latency and possibly fail to provide timely threat mitigation required in many applications. Machine learning models have proven to be effective techniques to detect these attacks by analysing large volumes of network traffic through patterns. Several studies have demonstrated the application of deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs) and graph neural networks (GNNs) to improve Intrusion Detection Systems' (IDS) efficiency.

Despite the improvements in machine learning-based security technologies, there are still some challenges. The skewed relationship between attack types in the training set can lead to biased models that fail to detect less common but equally dangerous attacks. The ability of attackers to adapt and devise new evasion detection techniques requires the creation of adaptive and self-evolving security systems. In addition, the computational demands of detecting threats in real-time should be considered to maintain both efficiency and accuracy effectively.

In this work, we study the use of low-complexity machine and deep learning techniques that can run directly on edge devices, ensuring low response times while maintaining high detection accuracy. In particular, we investigate shallow machine learning models and optimized decision trees, that are very suitable for low-power hardware platforms such as FPGAs, ARM-based processors, and microcontrollers.

Our study includes traditional classifiers such as Logistic Regression [2], Random Forest [3], and XGBoost [4], known for their efficiency and interpretability. We also examine Support Vector Machines [5] and K-Nearest Neighbors [6] to assess their ability to classify network anomalies effectively. On the deep learning side [7], we explore neural architectures that have demonstrated high performance in classification tasks. This includes Single-Layer Perceptron (SLP) and Multi-Layer Perceptron (MLP) [8] as lightweight neural network models [9], as well as more complex architectures like Deep Convolutional Neural Networks (DCNN) [10][11], Residual Networks (ResNet) [12], and Densely Connected Convolutional Networks (DenseNet) [13],[14].

We tested all these models based on their classification accuracy, precision, recall, F1-score, and inference time. To validate our approach, we used the CICIDS-2017 dataset, which provides a wide variety of network traffic scenarios, including normal and anomalous behaviors such as brute-force login attempts, botnet communication, and several types of DOS and DDoS attacks [15].

## 2 Related Works

Machine Learning has emerged as a solution for detecting and preventing DDoS attacks in IoT networks. Supervised and unsupervised learning techniques have been researched comprehensively to enhance the threat detection feature. Random Forest, Support Vector Machines (SVM), and Decision Trees classification algorithms, for example, have been able to achieve very high accuracy rates in the detection of malicious traffic patterns [16].

Deep learning techniques have used CNNs and long short-term memory (LSTM) networks in real-time attack detection [17]. Such models, even if being highly accurate for detection, are put at a disadvantage when it comes to lightweight implementation in IoT. For instance, the DDoSNet model [18] uses RNNs combined with autoencoders to detect DDoS attacks in Software-Defined Networking networks, addressing the limitations of traditional models in recognizing complex patterns of attack, but requiring a high computational cost.

Similarly, deployment of neural networks has proved effective in the detection of some of the types of DDoS attacks, as SYN Flooding, ACK Flooding, HTTP Flooding, and UDP Flooding, being suitable for real-world deployment [19]. Alfatemi et al. (2024) [20] proposed a hybrid approach that included residual deep neural networks and oversampling techniques of synthetic data for solving the class imbalance issue of their dataset. Pujol-Perich et al. (2021) [21] looked into the application of GNNs towards intrusion detection. With their approach that uses a graph perspective of network traffic flow, they achieved state-of-the-art results on CICIDS-2017. This reflects the ability that GNNs hold in revealing sophisticated network relationships which may be easily overlooked by other models. Belarbi et al. (2022) [22] proposed an intrusion detection mechanism using Deep Belief Networks with enhanced rates of detection on CICIDS-2017. The mechanism showed good performance, especially in detecting underrepresented categories of attacks, which are typically difficult for traditional IDS models to detect correctly.

Furthermore, feature selection is an important parameter for tuning the performance of intrusion detection systems that are based on ML. In [23], the CICIDS-2017 dataset has been employed to test such techniques, proving a high detection rate for different combinations of input features. In [24], the authors demonstrated that feature reduction with the help of Information Gain and Principal Component Analysis can significantly reduce the computational overhead without sacrificing the detection accuracy.

| Label | Wednesday | Friday | Total Count | Unique Count |
|---|---|---|---|---|
| BENIGN | 440,031 | 97,718 | 537,749 | 501,593 |
| DoS Hulk | 231,073 | - | 231,073 | 171,509 |
| DoS GoldenEye | 10,293 | - | 10,293 | 10,279 |
| DoS Slowloris | 5,796 | - | 5,796 | 5,289 |
| DoS Slowhttptest | 5,499 | - | 5,499 | 5,176 |
| Heartbleed | 11 | - | 11 | 11 |
| DDoS | - | 128,027 | 128,027 | 128,007 |
| **Total** | 692,703 | 225,745 | 918,448 | 822,864 |

**Table 1.** Overview of the attack distribution in the CICIDS2017 dataset, specifically for Wednesday and Friday. The table presents the number of benign and malicious network traffic samples recorded on each day. The "Total Count" column represents the sum of occurrences across both days, while the "Unique Count" column reports the number of distinct samples after removing duplicates across the datasets. This filtering ensures that redundant entries do not affect the model's learning process, improving generalization and robustness in intrusion detection.

## 3 Dataset

The CICIDS-2017 dataset [25] is a well-known dataset to evaluate IDS and Intrusion Prevention Systems. It was created by the Canadian Institute for Cybersecurity and aims to address current IDS datasets' limitations by providing realistic and labeled network traffic, both normal and anomalous. The dataset comprises several types of attacks such as Brute Force, DoS/DDoS, Port Scan, Bot, Web Attacks, Infiltration, and Heartbleed. Data was collected for five days, each day containing both normal and attack traffic, and was gathered on a network topology of firewalls, routers, switches, and a mix of Windows, Ubuntu, and macOS hosts.

In this study, only DoS/DDoS attacks, collected on Wednesday and Friday, are considered. The distribution of samples is described in Table 1, which identifies benign and malicious samples per scenario. Namely, Wednesday's data features different types of Denial of Service attacks: DoS Slowloris, DoS Slowhttptest, DoS Hulk, and DoS GoldenEye, which exploit server weaknesses to exhaust resources and make services unreachable. Friday's data consists of Distributed Denial of Service attacks, where hundreds of infected systems flood a target with an avalanche of traffic.

### 3.1 Preprocessing

We started the preprocessing of the dataset by combining the Wednesday and Friday data, making the instances unique and removing duplicates to eliminate redundancy and potential overfitting problems. The dataset contains features with different scales, we applied Min-Max Scaling, transforming all values into a range from 0 to 1 for uniformity and to avoid dominance of some features over others. The data was split into 80% training data (512 433 samples), 10%

validation data (64 054 samples), and 10% test data (64 054 samples) to allow for a good evaluation of the model's generalization. We also ensured that the class distribution was balanced, such that attack and benign traffic were nearly equal, preventing model bias toward the majority class.

## 4 Methodology

Once the dataset was preprocessed, we proceeded to the training stage, where we tested a broad variety of machine learning and deep learning models, both classic classifiers ( Logistic Regression, Random Forest, XGBoost) and neural network models like MLPs and CNN-based models. Each network has been studied based on its capability to correctly classify network traffic along with issues related to interpretability, generalization, and computational tractability.

For training the deep models, mini-batch gradient descent is applied along with the Adam optimizer and binary cross-entropy loss. For preventing overfitting and stable convergence, we implement an early stopping strategy based on the validation loss and stop the training when no improvement is observed in the predefined set of epochs.

In addition to classification accuracy, we carefully scrutinize inference latency per sample to provide information regarding the suitability of every model for real-time and low-resource environments. This intense analysis depicts the precision-performance tradeoff, enabling decision-making for practical cybersecurity applications.

We employed six machine learning models and five deep learning models in the study to learn network traffic patterns and detect cyber attacks. Each model has different characteristics in terms of complexity, explainability, and computational cost. We elaborate on each model below.

### 4.1 Machine Learning Models

We benchmarked a wide spectrum of machine learning models to distinguish between malicious traffic patterns and legitimate traffic patterns. The models differ in complexity, interpretability, and decision boundaries with complementary strengths depending on the structure of the input features.

Logistic Regression is a standard machine learning algorithm that, despite its simplicity, performs very well when the data is linearly separable. This model estimates the probability that some network flow belongs to the attack class using the sigmoid function

$$P(y = 1|x) = \frac{1}{1 + e^{-(w^T x + b))}} \tag{1}$$

where $x$ is the feature vector, $w$ is the learned weight vector, and $b$ is the bias. The training of this network is accomplished by minimizing the log loss via Batch Gradient Descent. This update rule computes the average gradient of the loss function using the entire training set at each step and then adjusts

the model parameters in the opposite direction of the gradient. With expressive input features, this model can achieve surprisingly good performance with low computational expense.

A more complex model is the Random Forest. It generates a decision structure as a collection of decision trees, each of them trained on a randomly sampled subset of the data, and combines their predictions through majority voting

$$\hat{y} = \text{majorVoting}(T_1(x), T_2(x), \ldots, T_n(x)) \tag{2}$$

with $T_i(x)$ being the output of the $i$-th decision tree. The ensemble approach improves robustness to noise and can learn non-linear relationships that are prevalent in the traffic features of DDoS data. In addition, the important features obtained from the forest can also yield insights into the most contributing elements for attack detection. To be able to deal with large-scale datasets effectively, we also considered the Stochastic Gradient Descent (SGD) algorithm. It differs from the batch gradient descent by updating the model parameters by using mini-batches of the training dataset

$$w_{t+1} = w_t - \eta L(w_t) \tag{3}$$

where $\eta$ is the learning rate and $L(w_t)$ is the loss function at iteration $t$ computed on a mini-batch of the entire dataset. Logistic loss was used in our scenario, which matched the binary classification paradigm. SGD requires careful hyperparameter tuning but, being scalable, is particularly suited for high-throughput network monitoring installations.

The K-Nearest Neighbors (KNN) algorithm offers a model-free solution. Every new input sample is classified by a majority vote of the $k$ closest examples in the training set

$$\hat{y} = \arg\max_c \sum_{i=1}^{k} \mathbb{1}(y_i = c) \tag{4}$$

where $\mathbb{1}$ is the indicator function and $y_i$ are the labels of the neighbors selected. Very important hyperparameters of this algorithm are the choice of $k$ and the distance metric function. While KNN may perform well in low-dimensional spaces, its prediction cost grows with the dataset size, which may limit real-time application in high-velocity DDoS detection pipelines.

We also included Extreme Gradient Boosting (XGBoost) in our comparison, due to its success in structured data tasks. The model builds an additive sequence of decision trees, where each new tree minimizes the residuals of the current ensemble

$$F_m(x) = F_{m-1}(x) + \gamma h_m(x) \tag{5}$$

where $F_m(x)$ is the ensemble at iteration $m$, $\gamma$ is the shrinkage factor, and $h_m(x)$ is the newest regression tree. XGBoost uses both $L_1$ and $L_2$ regularization, handles missing values, and is highly optimized for performance. Its ability to model faint feature interactions is useful in distinguishing between normal and abnormal traffic.

Finally, we examined Support Vector Machines, another classic machine learning model, which seeks the hyperplane that best separates the two classes by maximizing the margin which is expressed as

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{6}$$

where $x$ is the input feature vector, $w$ are the parameters of the model, and $b$ is the bias term. The optimization problem becomes

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i \tag{7}$$

For handling non-linear boundaries, a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ can be used. In detail, considering the DDoS detection task, we employed a Radial Basis Function (RBF) kernel of the form

$$K(x_i, x_j) = \exp\big(-\gamma \left\|x_i - x_j\right\|^2\big) \tag{8}$$

where parameter $\gamma$ controls how far the influence of a single training example reaches: a small $\gamma$ smooths the decision boundary, while a large $\gamma$ allows it to twist tightly around clusters of malicious flows. We decided to use this particular kernel because it can capture subtle relationships among features such as packet inter-arrival times, source IP entropy, and flow volume. The use of this RBF leads to the following decision function:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

where $\alpha_i$ are Lagrange multipliers and $y_i$ is the class of the i-th input sample.

### 4.2 Deep Learning Models

To further investigate the effectiveness of learning-based approaches to DDoS attack detection, we extended consideration to a range of deep learning models. These models provide a range of expressiveness, from shallow feedforward models to deeper convolutional models with residual connections.

The simplest neural network we examined is the Single-Layer Perceptron (SLP). Its output is computed as

$$y = \sigma\big(\mathbf{w}^T\mathbf{x} + b\big) \tag{10}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ the weight vector, $b$ the bias term, and $\sigma$ an activation function. In its original form, $\sigma$ is the binary step:

$$\sigma(x) = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{11}$$

Although this model can only separate data with a linear boundary, it serves as a useful baseline for neural classification and offers insight into whether data

representations are linearly separable. For our DDoS detection task, however, we replaced the step function with the Rectified Linear Unit (ReLU), defined by

$$\text{ReLU}(x) = \max(0,\, x) \tag{12}$$

which generally yields faster convergence and richer feature representations.

The Multi-Layer Perceptron (MLP) is a generalization of the SLP, which contains more hidden layers. Each hidden neuron calculates its activation as:

$$h_j = \sigma\left(\sum_{i=1}^{n} w_{ji} x_i + b_j\right) \tag{13}$$

where $w_{ji}$ is the weight from the $i$-th input to the $j$-th hidden neuron, and $b_j$ is the bias term. The output layer then projects the hidden representations via:

$$y_k = \sigma\left(\sum_{j=1}^{m} v_{kj} h_j + b_k\right) \tag{14}$$

where $v_{kj}$ links the $j$-th hidden neuron with the $k$-th output unit. In our DDoS detection problem, the MLP worked very well. It achieved very good classification performance at low inference latency with a good balance between model complexity and deployability in practice. Of all deep learning models that were experimented with, it worked best with common real-time network monitoring constraints.

To enhance spatial locality and hierarchical structure in network traffic features, we also examined the Dense Convolutional Networks (DenseNet). In this architecture, each layer receives as input the concatenation of the output of all preceding layers:

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \tag{15}$$

Here, $H_l$ is a composite function made of Convolution, ReLU activation, and Batch Normalization. As the network goes deeper, the number of feature maps in each layer increases linearly with respect to the growth rate $k$:

$$\text{Feature maps at layer } l = k_0 + k \times (l-1) \tag{16}$$

with $k_0$ feature maps in the first layer. Dense connectivity pattern aids in feature reuse and avoids the vanishing gradient problem in deep networks.

A more conventional methodology is used in the Deep Convolutional Neural Network (DCNN), in which every one of the convolutional layers accepts as input the output of the previous layer:

$$x_l = H_l(x_{l-1}) \tag{17}$$

The operation $H_l$ is usually comprised of Batch Normalization, a non-linear activation like ReLU, and a Convolution operation. Pooling layers are usually employed to downsample spatial dimensionality, followed by fully connected layers to generate the final classification. The DCNN architecture is capable of

extracting local correlations in input data and is thus appropriate to identify weak patterns in sequential or structured traffic representations.

Finally, we employed Residual Networks (ResNet) that are intended to enable the use of very deep models by adding identity-based skip connections. Any residual block can be represented as:

$$x_{l+1} = F(x_l) + x_l \tag{18}$$

where $F(x_l)$ is a non-linear transformation in the form of convolutions, batch normalization, and ReLU activations. The input $x_l$ is added by a skip connection, which enables the network to learn residual mappings and thus bypass the issue of degradation of deep networks. ResNet is able to effectively encode complicated feature hierarchies within traffic data.

## 5 Performance Metrics

The performance measures tested are accuracy, precision, recall, F1 score and inference time.

Accuracy is the ratio of correctly predicted instances over the total number of instances, and is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

where $TP$ is true positives, $TN$ is true negatives, $FP$ is false positives, and $FN$ is false negatives.

Precision measures the proportion of true positive predictions among all positive predictions made by the model:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{20}$$

Recall, also known as sensitivity or true positive rate, indicates the proportion of actual positives that were correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{21}$$

F1 Score is the harmonic mean of precision and recall, providing a balance between the two, especially useful in imbalanced datasets:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{22}$$

Lastly, inference time refers to the average time required by the model to make a prediction on a single sample. This metric is crucial to evaluate the feasibility of deploying a model in real-time applications, particularly on low-power IoT devices.

| Model | Accuracy | Precision | Recall | F1 Score | Inference (s) |
|---|---|---|---|---|---|
| Logistic Regression | 0.9901 | 0.9901 | 0.9901 | 0.9901 | **$1.11 \times 10^{-7}$** |
| Random Forest | **0.9992** | **0.9992** | **0.9992** | **0.9992** | $3.42 \times 10^{-6}$ |
| SGD Classifier | 0.9906 | 0.9906 | 0.9906 | 0.9906 | $1.95 \times 10^{-7}$ |
| KNN | **0.9989** | **0.9989** | **0.9989** | **0.9989** | $9.39 \times 10^{-5}$ |
| XGBoost | **0.9997** | **0.9997** | **0.9997** | **0.9997** | $7.56 \times 10^{-7}$ |
| SVM | 0.9972 | 0.9972 | 0.9972 | 0.9972 | $4.24 \times 10^{-4}$ |
| SLP | 0.9909 | 0.9906 | 0.9911 | 0.9909 | **$2.21 \times 10^{-9}$** |
| ResNet1D | 0.9973 | 0.9980 | 0.9966 | 0.9973 | $3.59 \times 10^{-7}$ |
| MLP | 0.9982 | 0.9983 | 0.9979 | 0.9981 | **$7.75 \times 10^{-9}$** |
| DCNN | 0.9985 | 0.9993 | 0.9979 | 0.9985 | $3.05 \times 10^{-4}$ |
| DenseNet | 0.9982 | 0.9986 | 0.9979 | 0.9982 | $2.94 \times 10^{-4}$ |

**Table 2.** Comparison of various Machine Learning and Deep Learning models for DoS/DDoS attack detection on the CICIDS2017 dataset. The comparison is made among models based on accuracy, precision, recall, F1-score, and inference time. The top three values for each parameter are shown in different colors: the top one is shown in red, the second top one in blue, and the third top one in green. It has been found that while XGBoost gives the highest accuracy, MLP offers the best balance of detection performance versus inference time. Light models such as SLP and Logistic Regression offer excellent computational efficiency and therefore are top of the list among real-time IoT-based security solution candidates.

## 6 Results

The results highlight that deep learning models such as DenseNet, DCNN, and MLP are the most accurate, with over 99%, making them very effective for network intrusion classification. However, DenseNet and DCNN possess high inference times, which limits their utility for real-time applications on resource-constrained IoT devices. MLP, being a deep model, delivers an acceptable inference time, which is high at increased efficiency.

Among the classic machine learning algorithms, XGBoost, with its reputation for achieving excellent accuracy (99.97%), has a comparatively high inference time, too. Similarly, algorithms like Random Forest and K-Nearest Neighbors, though accurate, are not computationally efficient in terms of speed. Light models like Logistic Regression and Single-Layer Perceptron (SLP) have very fast inference, with SLP being 99.09% accurate at an inference time of only $2.21 \times 10^{-9}$ sec/sample.

In both inference and accuracy, MLP is the most balanced option with the best trade-off for real-time deployment in IoT environments. Moreover, even if the confusion matrix, shown in Fig. 1, on the test set reveals 52 false negatives (approximately 0.16% of attack flows) and 67 false positives (around 0.21% of benign flows), such rates are effectively negligible: the model correctly flags over 99.8% of DDoS traffic while preserving more than 99.7% of legitimate flows. In an IoT context, where CPU and memory resources are limited and valuable, the MLP's nanosecond-scale inference ensures minimal impact on device performance, and any further reduction in error rates can be achieved through slight

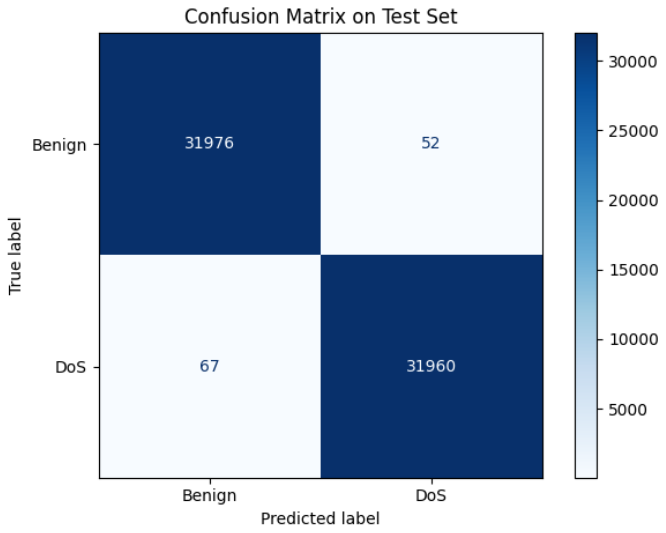


**Fig. 1.** Confusion matrix of the MLP model on the test set: True Positives (TP) = 39174, True Negatives (TN) = 31957, False Positives (FP) = 67, and False Negatives (FN) = 52.

threshold tuning or a lightweight secondary check on flows labeled benign, all without compromising its suitability for real-time deployment.

## 7 Conclusion

By presenting the trade-offs between detection quality and real-time inference requirements in intrusion detection in IoT networks, this study compares the performance of various machine learning and deep learning models to identify DDos/DoS attacks, highlighting the limitations of IoT deployment. Our study finds that machine learning models reach higher accuracy and are generally faster than deep learning methods, which suffer from higher computational costs, making them unsuitable for real-time deployment on IoT devices with limited resources. However, MLP are the best choice to implement, reaching a balance between high accuracy (99.82%) and inference time (second lowest inference time: $7.75 \times 10^{-9}$), being best suited for real-time intrusion detection. SLP has the minimum inference time, but lower accuracy, so it is the best for ultra-low latency IoT applications. Future developments will focus on incorporating advanced data preprocessing strategies, such as feature selection, dimensionality reduction, and automated noise filtering, to optimize input representations and further reduce computational overhead. Simultaneously, we intend to evaluate the proposed models within actual IoT settings to verify their efficacy and flexibility when faced with deployment limitations.

## Acknowledgments

his work was developed at the *is.Lab()* Intelligent Systems Laboratory at the Department of Computer, Control, and Management Engineering, Sapienza University of Rome. This paper have been partially funded by Italian Ministry of University and Research within the grant PRIN 2022 "ISIDE: Intelligent Systems for Infrastructural Diagnosis in smart-concretE", n. 2022S88WAY - CUP B53D2301318 and by the Age-It: Ageing Well in an ageing society project, task 9.4.1 work package 4 spoke 9, within topic 8 extended partnership 8, under the National Recovery and Resilience Plan (PNRR), Mission 4 Component 2 Investment 1.3 - Call for tender No. 1557 of 11/10/2022 of Italian Ministry of University and Research funded by the European Union - NextGenerationEU, CUP B53C22004090006. Valerio Ponzi carried out this research while enrolled in the Italian National PhD Program in Artificial Intelligence, coordinated by Sapienza University of Rome in collaboration with the same department.

## References

1. E. Džaferović, A. Sokol, A. Abd Almisreb, and S. M. Norzeli, "Dos and ddos vulnerability of iot: a review," *Sustainable Engineering and Innovation*, vol. 1, no. 1, pp. 43–48, 2019.
2. D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.
3. S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.
4. T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.
5. M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.
6. L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
7. V. Ponzi, S. Russo, A. Wajda, and C. Napoli, "A comparative study of machine learning approaches for autism detection in children from imaging data," in *CEUR Workshop Proceedings*, vol. 3398, p. 9 – 15, 2022.
8. M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.
9. E. Iacobelli, S. Russo, and C. Napoli, "A machine learning based real-time application for engagement detection," in *CEUR Workshop Proceedings*, vol. 3695, p. 75 – 84, 2023.
10. N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 international conference on communication and signal processing (ICCSP)*, pp. 0588–0592, IEEE, 2017.
11. S. Falciglia, F. Betello, S. Russo, and C. Napoli, "Learning visual stimulus-evoked eeg manifold for neural image classification," *Neurocomputing*, vol. 588, 2024.
12. B. Koonce and B. Koonce, "Resnet 50," *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pp. 63–72, 2021.

13. F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.
14. N. Brandizzi, A. Fanti, R. Gallotta, S. Russo, L. Iocchi, D. Nardi, and C. Napoli, "Unsupervised pose estimation by means of an innovative vision transformer," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13589 LNAI, p. 3 – 20, 2023.
15. J. Nazario, "Ddos attack evolution," *Network Security*, vol. 2008, no. 7, pp. 7–10, 2008.
16. A. A. Afuwape, Y. Xu, J. H. Anajemba, and G. Srivastava, "Performance evaluation of secured network traffic classification using a machine learning approach," *Computer Standards & Interfaces*, vol. 78, p. 103545, 2021.
17. A. Diro and N. Chilamkurti, "Leveraging lstm networks for attack detection in fog-to-things communications," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 124–130, 2018.
18. M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pp. 391–396, IEEE, 2020.
19. D. Tymoshchuk, O. Yasniy, M. Mytnyk, N. Zagorodna, and V. Tymoshchuk, "Detection and classification of ddos flooding attacks by machine learning method," *arXiv preprint arXiv:2412.18990*, 2024.
20. A. Alfatemi, M. Rahouti, R. Amin, S. ALJamal, K. Xiong, and Y. Xin, "Advancing ddos attack detection: A synergistic approach using deep residual neural networks and synthetic oversampling," *arXiv preprint arXiv:2401.03116*, 2024.
21. D. Pujol-Perich, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 111–117, 2022.
22. O. Belarbi, A. Khan, P. Carnelli, and T. Spyridopoulos, "An intrusion detection system based on deep belief networks," in *International Conference on Science of Cyber Security*, pp. 377–392, Springer, 2022.
23. D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, R. Budiarto, *et al.*, "Cicids-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.
24. S. K. Dash, S. Dash, S. Mahapatra, S. N. Mohanty, M. I. Khan, M. Medani, S. Abdullaev, and M. Gupta, "Enhancing ddos attack detection in iot using pca," *Egyptian Informatics Journal*, vol. 25, p. 100450, 2024.
25. R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.