



Paper Type: Original Article

## A Fast Internet of Things DDoS Attack Detection Method Using Deep Feedforward Networks

Vahide Babaiyan<sup>1,\*</sup>, Omid Bushehrian<sup>1</sup> , Reza Javidan<sup>1</sup> 

<sup>1</sup> Computer Engineering and IT Department, Faculty of Electrical and Computer Engineering, Shiraz University of Technology, Shiraz, Iran; v.babaiyan@sutech.ac.ir; bushehrian@sutech.ac.ir; javidan@sutech.ac.ir.

### Citation:

Received: 28 November 2024

Revised: 07 February 2025

Accepted: 22 April 2025

Babaiyan, V., Bushehrian, O., & Javidan, R. (2025). A fast internet of things DDoS attack detection method using deep feedforward networks. *Annals of process engineering and management*, 2(2), 101-111.

### Abstract


The increasing use of Internet of Things (IoT) devices has led to a surge in data traffic, which can be vulnerable to intentional denial-of-service (DoS) attacks that disrupt the intended Quality of Service (QoS). This paper presents a deep learning-based approach using Feedforward Neural Networks (FNNs) to detect Distributed Denial-of-Service (DDoS) attacks in IoT networks. We evaluated the performance of this approach on the IoT-23 dataset, which included captures of both malware-infected and benign IoT traffic. We conducted a comparative analysis between the FNN approach and three commonly used Machine Learning (ML) models, namely, Support Vector Machines (SVM), Random Forests (RFs), and Gradient Boosting (GRB). Our findings demonstrate that all methods achieve similar levels of accuracy. However, the FNN model distinguishes itself with significantly higher precision than the other models. Furthermore, our analysis revealed that FNN exhibits the shortest learning time among the considered models.


**Keywords:** Internet of things, Traffic classification, Supervised learning, Distributed denial-of-service attack, Internet of things-23.

## 1 | Introduction

The Internet of Things (IoT) refers to the growing network of interconnected devices that can communicate with each other and exchange data. It is an ever-increasing technology in many applications, such as smart cities, smart transportation systems, cloud computing, and smart medical care. This new platform provides communication between many heterogeneous mobiles or fixed systems in a way that no human intervention exists [1]. Due to the ever-increasing progress of the IoT and its diverse applications, a considerable volume of traffic has been created.

A Denial-of-Service (DoS) attack refers to a cyber-attack in which an assailant tries to disrupt the regular operations of a system or network by inundating it with an excessive amount of traffic or requests. Consequently, the system may become inaccessible or unresponsive to authorized users. IoT devices are particularly vulnerable to DoS attacks due to their limited processing power and memory, lack of proper

 Corresponding Author: v.babaiyan@sutech.ac.ir

 <https://doi.org/10.48314/apem.v2i2.33>



Licensee System Analytics. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

security measures, and the sheer number of devices connected to the network. The Distributed Denial of Service (DDoS) [2] in IoT networks describes an attack that intends to disrupt server availability by flooding the communication channel with fake requests from multiple IoT devices. Due to notable instances of server takedowns in recent times, safeguarding against IoT DDoS attacks has become a pressing research priority. IoT DDoS attacks can have serious consequences, such as disrupting critical infrastructure, causing financial losses, and compromising sensitive data. Therefore, there is a need for intelligent detection systems that can quickly and accurately identify and mitigate such attacks to ensure the security and reliability of IoT networks.

The growing number of IoT devices, diverse traffic patterns [3], [4] and malicious traffic caused by deliberate attacks necessitate the need to classify the traffic of IoT applications. Although the classification of network traffic has been one of the topics of interest since the early stages of the Internet, the growing development of various applications in the IoT has led to the demand for a more accurate classification of network traffic [5].

The traffic of an IoT network includes various data flows. In addition, there are also sources of traffic attacks that increase this traffic, which makes traffic classification a very challenging and important task. Therefore, security is one of the most fundamental areas that specifically benefit from network traffic classification [6]. Most of the research on network traffic classification has been published in Internet Protocol (IP) networks. Still, IoT traffic is very different from other types of network traffic because it follows an irregular pattern and unpredictable network behavior [7].

Researchers have proposed various methods and techniques to classify network traffic. The earliest solution relied on packet port-based classification but became ineffective for applications with port randomisation techniques [8]. Later, Deep Packet Inspection (DPI) of network traffic content filled the gap in the port-based solution [9]. However, this method also showed problems with encrypted network traffic. Recently, the use of Machine Learning (ML) algorithms has increased in the literature to classify IoT network traffic without needing access to port numbers or packet contents.

Our paper focuses on using supervised classification techniques to distinguish network traffic. Specifically, we concentrated on the deep learning approach used by Feedforward Neural Network (FNN) and assessed its effectiveness by applying three methods, namely Support Vector Machines (SVM), Random Forests (RFs), and Gradient Boosting (GRB). To enhance their performance, we utilised cross-validation to adjust the hyperparameter values of SVM, RF, and GRB. To evaluate the effectiveness of these techniques, we measured their accuracy, precision, recall, and F1\_score using the IoT-23 dataset, a novel dataset containing both benign and malicious network captures from various IoT devices.

The proposed methodology demonstrated superior performance in detecting DDoS attack patterns compared to standard ML methods. It exhibited comparable accuracy but stood out with significantly higher precision, especially in datasets with high dimensions and imbalanced data. Moreover, our approach achieved these results faster than traditional methods, making it more efficient.

Section 2 provides a concise review of related works. Section 3 describes the proposed methodology for addressing the IoT DDoS classification problem. Section 4 presents the method's implementation and evaluation. Finally, Section 5 concludes the paper and offers recommendations based on the findings.

## 2 | Related Work

Due to the severe danger that DDoS attacks pose to many IoT networks, numerous DoS detection and prevention methods have been suggested by researchers to classify network traffic. These classifications are compared regarding the accuracy, minimum loss, throughput, required computing resources, and speed [10]. These approaches can generally be categorised as packet port-based classification, DPI of network, and ML-based approaches. However, most existing methods have limitations that can affect their effectiveness in detecting such attacks. In this section, we discuss some of the negative points of the earliest solutions and highlight previous studies that have addressed them.

## 2.1 | Port-Based Methods

The earliest methodology for traffic classification is port-based classification, which uses the Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) headers of packets to gather information about port numbers. After determining the port number, the traffic is classified by comparing the allocated TCP/UDP port number with the extracted port number. This method is the fastest and simplest for traffic classification [11]. Port-based methods use TCP/UDP headers to identify traffic, but they have limitations due to modern applications using unregistered or randomly assigned port numbers to evade detection. Encryption and dynamic allocation of port numbers also reduce the effectiveness of this method, and tunnelling techniques further decrease its reliability. A study found that port-based classification achieved only 30-70% accuracy [11].

## 2.2 | Payload-Based Methods

Packet inspection inspects the communication flow [8] and identifies well-known patterns within packets. This method is called DPI and has shown reasonable detection rates. However, DPI has been found to have issues when dealing with encrypted network traffic.

## 2.3 | Machine Learning-Based Approaches

ML algorithms have become increasingly popular in the literature to classify IoT network traffic without relying on port numbers or packet contents. In these methods, the statistical characteristics of the data are extracted, which indicate the behaviour of a specific protocol or the flows of an application. Supervised, unsupervised, and semi-supervised ML methods have been effective in traffic classification, although each has strengths and weaknesses. Limitations of port-based and payload-based methods can be addressed more effectively using ML-based techniques [12].

This section will present research on utilising ML methods to classify IoT network traffic.

Khedkar and AroulCanessane [13] used the SVM algorithm to distinguish between normal and malicious traffic for IoT network classification. A confusion matrix, Receiver Operating Characteristic (ROC) curve, and classification report were utilised to evaluate the model's effectiveness. Santos et al. [14] utilised a supervised ML algorithm called RF along with content inspection of packets to detect network traffic. The study employed cross-validation and hold-out techniques, and the outcome displayed an accuracy of approximately 99%. Bismukhamedov and Nadeev [15] examined the efficacy of straightforward models, such as Logistic Regression, SVM with a linear kernel, and Decision Tree, in the context of multiclass classification of IoT traces, considering careful feature engineering for real-world deployment purposes. Kumar et al. [16] compared ML algorithms using useful features extracted from IoT network traffic. A public dataset was utilised for this purpose. The researchers employed well-known ML algorithms to classify IoT traffic, and the effectiveness of these algorithms was evaluated comparatively based on criteria such as classification accuracy, speed, and training time.

Alzahrani and Alzahrani [3], Tahaei et al. [4], and Kumar et al. [16] had additional research, including review studies, which have thoroughly explored a diverse set of datasets and a broad spectrum of ML models. Shaaban et al. [17] introduced a Convolutional Neural Network (CNN) technique to accurately identify and classify DDoS traffic as either normal or malicious. The approach achieved 99% accuracy across two datasets and is compared to other classification algorithms like decision trees, SVMs, K-nearest neighbours, and neural networks. Stoian [18] investigated the effectiveness of ML algorithms in detecting anomalies within IoT networks to enhance their security. RF, Naive Bayes (NB), Multi-Layer Perceptron (MLP), SVM, and AdaBoost (ADA) were compared. Among these algorithms, the RF algorithm demonstrated the highest performance, achieving an accuracy of 99.5%. A new approach called "Deep Defense" was proposed by Yuan et al. [19] for detecting DDoS attacks using deep learning. Experimental results showed that the model outperformed traditional ML models, reducing the error rate from 7.517% to 2.103% in a larger dataset.

Aswad et al. [20] proposed a DDoS detection model by combining three deep learning algorithms. CICIDS2017 was used as an evaluation dataset. The proposed method achieves exceptional accuracy, 99.76%, and precision, reaching 98.90%.

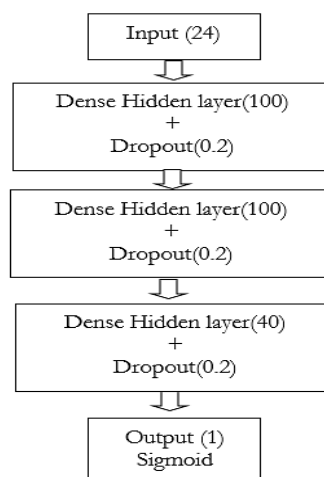
The literature proposes various methods for classifying IoT traffic using various ML algorithms. However, the accuracy of these algorithms depends on factors such as the data generated from different IoT devices, the features extracted from network traffic, and the location where the IoT system is deployed. Additionally, manual selection of features and ML algorithms can lead to errors. Consequently, there is a need to thoroughly examine network traffic characteristics and suitable ML algorithms to achieve accurate and optimised IoT traffic classification.

Our suggested approach builds on these previous studies by utilising ensemble methods and cross-validation techniques to optimise hyperparameters for IoT DoS attack detection. By doing so, we aim to overcome the limitations of existing processes and achieve higher accuracy, precision, recall, and F1 score in detecting DoS attacks on IoT networks.

### 3 | Proposed Methodology

The proposed methodology for addressing the IoT DDoS classification problem uses a FNN. FNNs [21] are a widely used artificial neural network that processes information in a one-directional flow, from the input layer through hidden layers to the output layer. FNNs consist of interconnected nodes (Neurons) organised into layers, including input, hidden, and output layers. They excel at pattern recognition, regression analysis, and classification tasks. FNNs automatically learn and extract complex patterns from data by adjusting the weights and biases of the neurons during training, aiming to minimise the difference between predicted and actual outputs. This learning process employs algorithms like back-propagation to update network parameters based on calculated gradients.

In our study, our model was developed using the Keras library. The FNN architecture used in this study consisted of multiple layers of densely connected nodes, shown in *Fig. 1*. The model starts with an input layer of 24 nodes, followed by three hidden layers with 100, 100, and 40 nodes, respectively. Each hidden layer is activated using the sigmoid function and includes a dropout layer with a dropout rate of 0.2, which helps prevent overfitting. The final layer of the FNN model has a single node activated using the sigmoid function, representing the binary classification output. The model is compiled using the Adam optimiser, and the loss function used is binary cross-entropy. The model's accuracy is evaluated as a metric during training and evaluation.



**Fig. 1.** Feedforward neural network architecture.

Here's an outline of the steps we follow to complete the classification task. We will explain these steps in detail in the next section.

- I. Preprocessing of the data
- II. Splitting the data
- III. Training and fitting the model
- IV. Evaluating the model on test data

## 4 | Implementation and Evaluation

In this study, we introduced a deep learning-based approach utilising an FNN. We integrated three commonly used supervised classification techniques, namely SVM, GRB, and RF, which are known for their high accuracy. The FNN, RF, GRB, and SVM models were implemented and trained using appropriate libraries, enabling us to evaluate their effectiveness. The following steps were taken in our study:

### 4.1 | Preprocessing the Data

Our study utilised the IoT-23 dataset [22], which consists of network traffic data from IoT devices. The dataset included 20 captures of malware traffic and 3 of benign traffic. We filtered the data to select specific labels, namely 'DDoS' and 'Benign'. Data preprocessing involved handling missing values, scaling or normalising features, and encoding categorical variables if necessary. The labels were mapped to numeric values, and the input data was normalised using a MinMaxScaler. These steps prepared the filtered samples for classification.

### 4.2 | Splitting the Data

The dataset was split into training, validation, and test sets to evaluate the model's performance and ensure its generalisation capability. This splitting strategy assigns 70% of the data for training the model, 15% for evaluating the model's performance during development or hyperparameter tuning, and the remaining 15% for the final evaluation of the model on unseen data. This approach allows for a thorough assessment of the model's performance on different datasets and ensures a more reliable estimation of its generalisation capabilities. We use the train-test-split function from sklearn. Model-selection to divide the preprocessed data into training and testing sets. The random-state parameter is set to 42 for consistent reproducibility of the split.

### 4.3 | Training and Fitting the Model

The FNN model was trained using the fit function with a batch size of 32 and 20 epochs. The model's weights and biases were adjusted using the Adam optimiser to minimise the binary cross-entropy loss function. A validation dataset was used to evaluate the model's performance after each epoch. The execution time for fitting the model was measured using the time module, providing insights into the computational cost of training. The fit function was crucial in optimising the model's parameters and improving its classification performance.

### 4.4 | Evaluate the Model on Test Data

The modelling tasks were conducted on a desktop computer with an Intel(R) Core(TM) i7-1065G7 CPU, running at a base frequency of 1.30 GHz and a maximum turbo frequency of 1.50 GHz. The computer has 8.00 GB of installed RAM, with 7.81 GB usable for our modelling tasks. For the programming aspect, we used Python as the primary tool for implementing our models. We chose Jupyter Notebook as our Integrated Development Environment (IDE) throughout development. By utilising the computational power of our desktop computer, leveraging the capabilities of the Python programming language, and utilising Jupyter Notebook as our IDE, we constructed a robust modelling framework for our environmental system.

The algorithm's performance is measured using different metrics [23], [24]. Five commonly used metrics assessed the classifier's performance: accuracy, precision, recall, and F-measure. The confusion matrix, a widely adopted method, was utilised to measure the accuracy and correctness of the model. This matrix comprises two dimensions, "Actual" and "Predicted," with a set of "Classes" present in both dimensions. Although the confusion matrix is not a standalone evaluation measure, most criteria are defined based on its values. Four terms associated with the confusion matrix were identified: True positives (Predicted and actual classes are both true), True negatives (Predicted and actual classes are both false), False positives (Predicted class is True, but the actual class is False), and False negatives (Predicted class is false but the actual class is true). These terms provide valuable insights for evaluating the classifier's performance in various classification scenarios.

Accuracy in classification problems is the number of correct predictions of the model in proportion to the total number of predictions made. Accuracy is a good measure when the target variable classes in the data are almost balanced. When the target variable classes are unbalanced, accuracy [25] should never be used as an evaluation criterion. The accuracy calculation method is stated in *Eq. (1)*.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}). \quad (1)$$

Precision, a metric used to evaluate performance, is calculated by dividing the number of correctly predicted positives by the total predicted positives. It is especially valuable when minimising false positives, which is more important than false negatives. In IoT attack classification, precision is crucial in reducing false positives. False positives occur when normal instances are mistakenly identified as attacks, resulting in unnecessary alerts or actions that consume resources and cause disruptions. Maintaining high precision makes the identified attacks more accurate and reliable, decreasing the likelihood of false alarms. This study highlights the significance of this specific metric.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}). \quad (2)$$

Recall (Sensitivity) explains how many positive cases we could predict correctly with our model. It is a useful metric in cases where a False Negative is of higher concern than a False Positive. Recall for a label is defined as the number of true positives divided by the total number of actual positives.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}). \quad (3)$$

F1 Score gives a mixed impression of the precision and recall measures. This metric reaches its maximum value when precision is equal to recall. The F1 score is the harmonic mean of precision and recall.

$$\text{F1 - Score} = (2 * \text{Precision} * \text{recall}) / (\text{precision} + \text{recall}). \quad (4)$$

We compared the performance of the models based on the evaluation metrics and created visualisations and summary tables to compare the metrics across the models.

We employed the "GridSearchCV" technique [26] to identify the optimal parameter values. This technique extensively searches a predefined grid of parameter values. This technique systematically combines the model with the selected parameters and their specified ranges. We determined the most favourable parameter values by utilising the "grid-cv" method, resulting in improved efficiency and more accurate predictions. We adjusted their parameters for the fundamental classification methods (RF, GRB, and SVM) according to *Table 1*, which presents the best values obtained through this process.

**Table 1. Hyperparameter extracted values.**

Model	Hyperparameter Grids	Best Hyperparameters
SVM	{'C': [0.1, 1, 10, 100], 'kernel': ['linear']}, {'C': [0.1, 1, 10, 100], 'kernel': ['poly'], 'degree': [2, 3, 4]}, {'C': [0.1, 1, 10, 100], 'kernel': ['rbf'], 'gamma': [0.1, 1, 10]}, {'C': [0.1, 1, 10, 100], 'kernel': ['sigmoid'], 'gamma': [0.1, 1, 10], 'coef0': [0.1, 1, 10]}	'C': 0.1, 'kernel': 'linear'



**Table 1. Continued.**

Model	Hyperparameter Grids	Best Hyperparameters
RF	{ 'n_estimators': [100, 200, 300], 'max_depth': [None, 5, 10], 'max_features': ['sqrt', 'log2'], 'min_samples_split': [2, 5, 10], }	'max_depth': None, 'max_features': 'sqrt', 'min_samples_split': 2, 'n_estimators': 100
GRB	{ 'learning_rate': [0.1, 0.01], 'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 7], 'subsample': [0.8, 1.0] }	'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100, 'subsample': 0.8

The parameters provided to the function are crucial and can significantly influence the method's predictive performance. After setting the parameters to appropriate values, we calculate the scores of these models. The results obtained from the models are presented in *Table 2*.

**Table 2. Performance evaluation of basic classifier.**

Model	Accuracy	Precision	Recall	F1_Score
FNN	0.90	0.99	0.80	0.89
SVM	0.90	0.91	0.90	0.89
RF	0.90	0.91	0.90	0.89
GRB	0.90	0.91	0.90	0.89

Compared to existing methods in the literature [13], when we utilise ensemble methods like RF and GRB and apply cross-validation techniques to optimise hyperparameters for IoT DDOS attack detection, we achieve higher performance and reduce overfitting.

Based on the findings, the SVM, RF, and GRB approaches demonstrate comparable performance across various evaluation metrics, including accuracy, precision, recall, and F1 score. These metrics are not calculated explicitly for a particular class (e.g., attack or benign) but provide an overall evaluation of the model's performance in classifying both classes. All these methods achieve high accuracy scores, as presented in *Table 1*. In contrast, the FNN model shows similar accuracy but significantly higher precision than the other models (0.99 versus 0.91). This difference suggests that the FNN model accurately identifies true positives while minimizing false positives. This specific metric is important, especially when prioritising precise predictions for the given problem.

Considering the lack of discernible performance differences based on the given metrics, it is advisable to consider additional factors such as computational complexity, implementation simplicity, and specific problem requirements when deciding on the most suitable approach. In addition to evaluating the metrics, we also calculated the execution time of the examined methods, considering both the tuning time and fitting time. *Fig. 2* illustrates the results, visually representing the execution time of each technique.

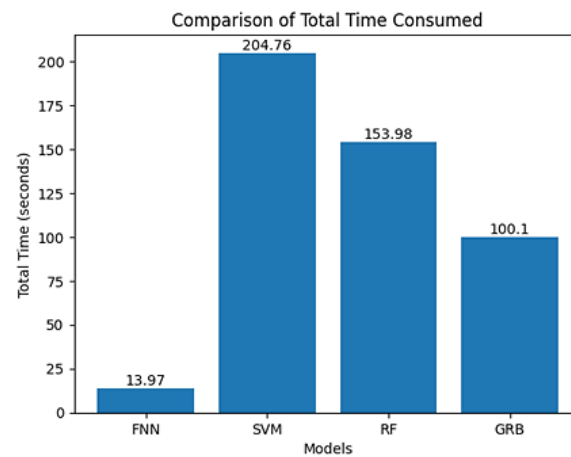


Fig. 2. Comparison of total time consumed.

Considering the FNN model's precision score of 0.99 and its relatively lower time consumption of 13.97 seconds, it becomes evident that the model demonstrates exceptional precision in generating accurate predictions while maintaining efficiency. These findings highlight the model's proficiency in accurately detecting true positives and effectively minimising false positives. The superior precision of the FNN model makes it a strong candidate for applications where precise predictions are of utmost importance. Moreover, the relatively shorter time required for training and evaluation adds to the practicality and efficiency of the FNN model. Therefore, the FNN model presents a favourable choice in scenarios where high precision and computational efficiency are essential considerations.

As our primary focus is evaluating the FNN method, we have observed that its performance metrics are highly favourable, and it also exhibits faster computation compared to other methods. To assess the accuracy and loss of both the training and validation sets and to identify any signs of overfitting, we have provided visual representations in *Fig. 3* and *Fig. 4*. These figures demonstrate that the model does not suffer from the issue of overfitting.

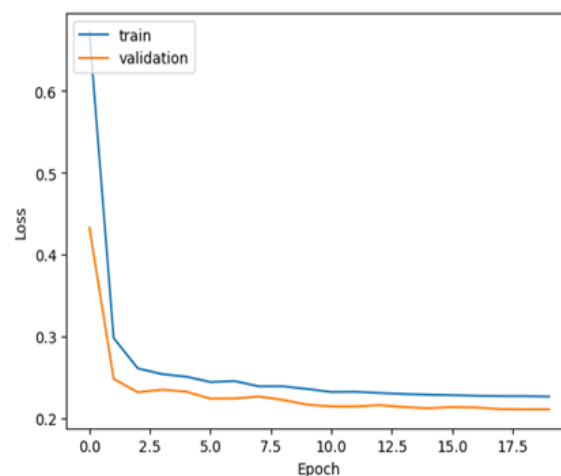


Fig. 3. The accuracy of the feedforward neural network-based model.



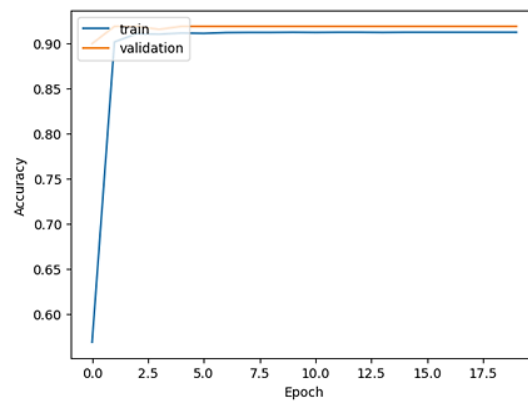


Fig. 4. The loss of the feedforward neural network-based model.

These results demonstrate that our FNN-based approach can detect DDoS attacks in IoT networks in binary classification tasks.

## 5 | Conclusion

Network traffic classification aims to identify applications or services used in a network. This process offers various advantages, such as implementing Quality of Service (QoS) for real-time application traffic, restricting specific applications, enforcing regulations, and detecting malicious activity. This study utilizes a supervised learning classifier and statistical features to overcome the limitations of Port-based methods and DPI. The IoT-23 dataset is employed to analyze both DDoS attacks and benign data. Upon evaluating different ML methods, including SVM, RF, GRB, and FNN, it is observed that these methods demonstrate comparable levels of accuracy. However, the proposed FNN model stands out with significantly higher precision than the other models. The higher precision of the FNN model indicates its effectiveness in identifying true positives while minimizing false positives. When selecting the most suitable method, considerations should also be given to factors such as computational complexity, implementation simplicity, and the specific requirements of the problem. Analyzing the execution time, including tuning time and fitting time, provides valuable insights into the computational efficiency of the methods. While SVMs can become computationally expensive with large datasets or high-dimensional feature spaces, FNNs can efficiently process and train on such data. Furthermore, FNNs can generalize well to unseen data and are less susceptible to overfitting when sufficient training data and regularization techniques are employed.

In summary, a comprehensive assessment of performance metrics, computational complexity, and practical considerations is crucial when choosing the most appropriate method for network traffic classification. This comprehensive approach ensures optimal results for the specific problem at hand.

## References

- [1] Zafar, S., Jangsher, S., Bouachir, O., Aloqaily, M., & Othman, J. Ben. (2019). QoS enhancement with deep learning-based interference prediction in mobile IoT. *Computer communications*, 148, 86–97. <https://doi.org/10.1016/j.comcom.2019.09.010>
- [2] Vishwakarma, R., & Jain, A. K. (2020). A survey of DDoS attacking techniques and defence mechanisms in the IoT network. *Telecommunication systems*, 73(1), 3–25. <https://doi.org/10.1007/s11235-019-00599-z>
- [3] Alzahrani, R. J., & Alzahrani, A. (2021). Survey of traffic classification solution in IoT networks. *International journal of computer applications*, 183(9), 37–45. <https://doi.org/10.5120/ijca2021921392>
- [4] Tahaei, H., Afifi, F., Asemi, A., Zaki, F., & Anuar, N. B. (2020). The rise of traffic classification in IoT networks: A survey. *Journal of network and computer applications*, 154, 102538. <https://doi.org/10.1016/j.jnca.2020.102538>

- [5] Finsterbusch, M., Richter, C., Rocha, E., Muller, J. A., & Hanssger, K. (2014). A survey of payload-based traffic classification approaches. *IEEE communications surveys & tutorials*, 16(2), 1135–1156. <https://doi.org/10.1109/SURV.2013.100613.00161>
- [6] Al Khater, N., & Overill, R. E. (2015). Network traffic classification techniques and challenges. *2015 tenth international conference on digital information management (ICDIM)* (pp. 43–48). IEEE. <https://doi.org/10.1109/ICDIM.2015.7381869>
- [7] Shahid, M. R., Blanc, G., Zhang, Z., & Debar, H. (2018). IoT devices recognition through network traffic analysis. *2018 IEEE international conference on big data (Big data)* (pp. 5187–5192). IEEE. <https://doi.org/10.1109/BigData.2018.8622243>
- [8] Moore, A., Zuev, D., & Crogan, M. (2005). *Discriminators for use in flow-based classification*. <https://www.researchgate.net/publication/243787961>
- [9] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE access*, 5, 18042–18050. <https://doi.org/10.1109/ACCESS.2017.2747560>
- [10] Azab, A., Khasawneh, M., Alrabaa, S., Choo, K.-K. R., & Sarsour, M. (2022). Network traffic classification: Techniques, datasets, and challenges. *Digital communications and networks*, 10(3), 676–692. <https://doi.org/10.1016/j.dcan.2022.09.009>
- [11] Qi, Y., Xu, L., Yang, B., Xue, Y., & Li, J. (2009). Packet classification algorithms: From theory to practice. *IEEE INFOCOM 2009* (pp. 648–656). IEEE. <https://doi.org/10.1109/INFCOM.2009.5061972>
- [12] Shafiq, M., Yu, X., Laghari, A. A., Yao, L., Karn, N. K., & Abdessamia, F. (2016). Network traffic classification techniques and comparative analysis using machine learning algorithms. *2016 2nd IEEE international conference on computer and communications (ICCC)* (pp. 2451–2455). IEEE. <https://doi.org/10.1109/CompComm.2016.7925139>
- [13] Khedkar, S. P., & AroulCanessane, R. (2020). Machine learning model for classification of IoT network traffic. *2020 fourth international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)* (pp. 166–170). IEEE. <https://doi.org/10.1109/I-SMAC49090.2020.9243468>
- [14] Santos, M. R. P., Andrade, R. M. C., Gomes, D. G., & Callado, A. C. (2018). An efficient approach for device identification and traffic classification in IoT ecosystems. *2018 IEEE symposium on computers and communications (ISCC)* (pp. 304–309). IEEE. <https://doi.org/10.1109/ISCC.2018.8538630>
- [15] Bikmukhamedov, R. F., & Nadeev, A. F. (2019). Lightweight machine learning classifiers of iot traffic flows. *2019 systems of signal synchronization, generating and processing in telecommunications (SYNCHROINFO)* (pp. 1–5). IEEE. <https://doi.org/10.1109/SYNCHROINFO.2019.8814156>
- [16] Kumar, R., Swarnkar, M., Singal, G., & Kumar, N. (2021). IoT network traffic classification using machine learning algorithms: An experimental analysis. *IEEE internet of things journal*, 9(2), 989–1008. <https://doi.org/10.1109/JIOT.2021.3121517>
- [17] Shaaban, A. R., Abd-Elwanis, E., & Hussein, M. (2019). DDoS attack detection and classification via convolutional neural network (CNN). *2019 ninth international conference on intelligent computing and information systems (ICICIS)* (pp. 233–238). IEEE. <https://doi.org/10.1109/ICICIS46948.2019.9014826>
- [18] Stoian, N. A. (2020). *Machine learning for anomaly detection in iot networks: Malware analysis on the iot-23 data set*. [Thesis]. <https://B2n.ir/fm4029>
- [19] Yuan, X., Li, C., & Li, X. (2017). Deepdefense: Identifying DDoS attack via deep learning. *2017 IEEE international conference on smart computing (smartcomp)* (pp. 1–8). IEEE. <https://doi.org/10.1109/SMARTCOMP.2017.7946998>
- [20] Aswad, F. M., Ahmed, A. M. S., Alhammadi, N. A. M., Khalaf, B. A., & Mostafa, S. A. (2023). Deep learning in distributed denial-of-service attacks detection method for Internet of Things networks. *Journal of intelligent systems*, 32(1), 20220155. <https://doi.org/10.1515/jisys-2022-0155>
- [21] Upadhyay, Y. (2019). Introduction to feedforward neural networks. *Towards data science*, 7.
- [22] Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic. <http://doi.org/10.5281/zenodo.4743746>

- [23] Sunasra, M. (2017). *Performance metrics for classification problems in machine learning*. <https://medium.com/@MohammedS/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b>
- [24] Jia, Y., Zhong, F., Alrawais, A., Gong, B., & Cheng, X. (2020). Flowguard: An intelligent edge defense mechanism against IoT DDoS attacks. *IEEE internet of things journal*, 7(10), 9552–9562. <https://doi.org/10.1109/JIOT.2020.2993782>
- [25] Azab, A., Layton, R., Alazab, M., & Oliver, J. (2014). Mining malware to detect variants. *2014 fifth cybercrime and trustworthy computing conference* (pp. 44–53). IEEE. <https://doi.org/10.1109/CTC.2014.11>
- [26] Vieira, S., Garcia-Dias, R., & Pinaya, W. H. L. (2020). A step-by-step tutorial on how to build a machine learning model. In *Machine learning* (pp. 343–370). Elsevier. <https://doi.org/10.1016/B978-0-12-815739-8.00019-5>