Journal of
Sensor and Actuator Networks

MDPI

Article

Eye-Net: A Low-Complexity Distributed Denial of Service Attack-Detection System Based on Multilayer Perceptron

Ramzi Khantouchi [1], Ibtissem Gasmi [1] and Mohamed Amine Ferrag [2,*]

[1] Computer Science and Applied Mathematics Laboratory, Chadli Bendjdid El Tarf University, El Tarf 36000, Algeria; r.khantouchi@univ-eltarf.dz (R.K.); gasmi-ibtissem@univ-eltarf.dz (I.G.)
[2] Department of Computer Science, Guelma University, Guelma 24000, Algeria
* Correspondence: ferrag.mohamedamine@univ-guelma.dz

Abstract: Distributed Denial of Service (DDoS) attacks disrupt service availability, leading to significant financial setbacks for individuals and businesses. This paper introduces Eye-Net, a deep learning-based system optimized for DDoS attack detection that combines feature selection, balancing methods, Multilayer Perceptron (MLP), and quantization-aware training (QAT) techniques. An Analysis of Variance (ANOVA) algorithm is initially applied to the dataset to identify the most distinctive features. Subsequently, the Synthetic Minority Oversampling Technique (SMOTE) balances the dataset by augmenting samples for under-represented classes. Two distinct MLP models are developed: one for the binary classification of flow packets as regular or DDoS traffic and another for identifying six specific DDoS attack types. We store MLP model weights at 8-bit precision by incorporating the quantization-aware training technique. This adjustment slashes memory use by a factor of four and reduces computational cost similarly, making Eye-Net suitable for Internet of Things (IoT) devices. Both models are rigorously trained and assessed using the CICDDoS2019 dataset. Test results reveal that Eye-Net excels, surpassing contemporary DDoS detection techniques in accuracy, recall, precision, and F1 Score. The multiclass model achieves an impressive accuracy of 96.47% with an error rate of 8.78%, while the binary model showcases an outstanding 99.99% accuracy, maintaining a negligible error rate of 0.02%.

Keywords: neural network quantization; energy efficiency; low complexity; DDoS attacks; internet of things (IoT)

Citation: Khantouchi, R.; Gasmi, I.; Ferrag, M.A. Eye-Net: A Low-Complexity Distributed Denial of Service Attack-Detection System Based on Multilayer Perceptron. *J. Sens. Actuator Netw.* **2024**, *13*, 45. https://doi.org/10.3390/jsan13040045

Academic Editor: Chengwen Luo

Received: 10 July 2024
Revised: 6 August 2024
Accepted: 8 August 2024
Published: 12 August 2024

1. Introduction

Cyberattacks are hostile actions intended to harm computer systems by obtaining unauthorized access. Numerous threats exist, including click fraud, computer viruses, denial of service, malvertising/ad fraud, etc. [1]. DDoS attacks are the most common and damaging due to their straightforward yet potent offensive abilities. They are seen as a serious threat to the current Internet community [2,3]. They utilize a network of infected computers known as botnets to overload service providers with excessive requests to prevent legitimate users from accessing a target system or services [4].

DDoS attacks are becoming increasingly serious and harmful to any business or service in today's cyber world. As reported by NETSCOUT's ATLAS Security Engineering & Response Team (ASERT), attackers executed about 2.9 million DDoS threats in the initial trimester of 2021, an increase of 31% compared to the same period in 2020 [5]. If a suitable protection mechanism is not used, a DDoS victim may lose all or part of their services and files because the networks or processing capabilities cannot function normally [2].

Internet-based applications generate mixed packet chains as the signal moves from the original port to its destination. So, it is relevant to use information from network traffic flows to detect anomalous events when they happen to carry out prompt corrective actions. Therefore, several approaches using machine learning (ML) techniques were

proposed to detect DDoS attacks, such as Naive Bayes [6], Decision Tree [7], Random Forest [8], and Support Vector Machine [9]. However, traditional ML methods fail to deal with a huge amount of data due to their limited capacity to learn the features. They may find similarities in known threats rather than detect anomalous behavior for unidentified malignant attacks [10]. Recently, several researchers have been motivated by the exceptional success of deep learning (DL), and they have used DL techniques for intrusion detection, including Multilayer Perceptron, convolutional neural networks, and recurrent neural networks [5,11,12].

The authors of [13] studied various approaches based on classical machine learning techniques employed in attack detection. They concluded that the existing methods fail to identify sophisticated attacks in large-scale network environments and have several challenges that can cause poor classification results. The authors also confirmed that deep learning (DL) completely surpasses traditional methods.

Many studies of intrusion detection suffer from several issues, including the class imbalance that arises once the amount of intrusion samples is much less than the number of normal ones. Consequently, the models prefer the majority class and cannot obtain sufficient details about the minority classes from small amounts of data [14]. The overfitting problem occurs when the model learns noise and irrelevant features from the training data; this reduces its generalization ability on new data [15]. Most of the proposed models do not consider current datasets, which makes them unable to study the most recent DoS/DDoS attacks [16]. Moreover, the trade-off between accuracy, energy efficiency, and time inference is often not considered, which makes existing models unpractical to use, especially in IoT environments.

To solve these issues, the present study proposes a novel intrusion detection method called Eye-Net, which can accurately capture different types of attacks. Eye-Net consists of four components: the data selection component, the data generation component, the detection component, and the quantization component. The feature selection is based on ANOVA [17]. It is applied to remove redundant and irrelevant data, alleviate the overfitting issue, and reduce the complexity of the time needed to build the model. In the data generation module, the SMOTE technique [18] is used to handle the class imbalance problem. ANOVA and SMOTE techniques are applied only in the training stage. The detection component identifies intrusion activities and classifies network traffic using Multilayer Perceptron (MLP). On the other hand, MLP-based techniques must take into account the resource constraints that usually occur in IoT environments. IoT devices generally have limited computing power, memory, and energy sources. Consequently, this study introduces a quantization-aware training technique to quantize the weights and biases of the model from 32-bit floating-point precision to 8-bit fixed-point precision (INT8) to reduce memory consumption, improve the time complexity of the model, and improve the inference time, as well as make the model more energy-efficient. It also preserves the model accuracy to maintain powerful intrusion detection capabilities for real-time security by quantizing the weights and biases in each layer separately, starting with the first layer. After quantizing each layer, we keep its weights and biases untrainable and retrain the neural network to update the weights and biases of the other layers to correct the errors caused by the quantization operation in that layer.

The main contributions of this study include the following:

1. We propose a hybrid deep learning model that combines feature selection, an oversampling strategy, and MLP to detect DDoS attacks with remarkable improvements and optimum architecture.
2. We introduce a quantization-aware training algorithm to quantize the model weights and biases to INT8, significantly improving energy efficiency, memory usage, computational complexity, and inference time.
3. We evaluate the proposed method in terms of binary classification and multiclass classification using the CICDDoS2019 dataset.

4. We incorporate quantization-aware training, feature selection, and data balancing techniques to enhance the efficiency, accuracy, and time inference of the MLP model in detecting DDoS attacks on IoT devices.

The rest of this paper is structured as follows: Section 2 summarizes the literature review for intrusion detection. Section 3 describes the proposed Eye-Net model. The dataset, the evaluation metrics, and the experimental results are highlighted in Section 4. Section 5 discusses the limitations of Eye-Net. Finally, in the last section, our conclusions and future work are presented. Table 1 summarizes the notations used in the paper.

Table 1. Notations.

Notation	Description	Notation	Description	Notation	Description
D	The model	θ	The model parameters	l_{idx}	Layer index
θ_Q	The quantized parameters of the model	S	Scaling factor	b	Bias vector
A	Accumulator	W	Weight matrix	P	Processing unit

2. Related Work

The detection and mitigation of DDoS attacks present a significant challenge to network security and have been an active area of research [5]. Over the years, a range of DDoS detection methods has been proposed, including rule-based approaches and more advanced machine learning techniques [19–23].

The authors of [9] used the Grid Search Cross-Validation exhaustive parameter search method and the Radial Basis Function Kernel of the Support Vector Machine for DDoS attack detection in SDN integrated vehicular networks. They demonstrated that the proposed model performed better in classification accuracy and generalization than several algorithms.

The study in [16] presented a modular and flexible SDN-based architecture to detect DoS/DDoS attacks with ML and DL models. Multiple techniques were evaluated separately using CICDDoS2017 and CICDDoS2019 datasets to determine which performs better under different attack types and conditions. The experiments demonstrated accuracy above 99% in classifying unseen data. The authors also evaluated their solution in a simulated environment using Mininet and the ONOS controller. They concluded that DL provided better detection rates than ML models. Furthermore, the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) kept the most significant detection rates in the inline model evaluation. However, there was a difference between the testing network topology and the network configuration employed during the training phase of the models, resulting in a slight decrease in their performance. Moreover, complex GRU models were used, which led to high computational costs.

The authors of [24] combined sequential feature selection with MLP to detect DDoS attacks. During the training phase, they identified the optimal features and created a feedback mechanism to reconstruct the detector when significant detection errors are dynamically perceived. The method was tested on the NSL-KDD dataset and compared with some papers in the literature. The experiment showed that the accuracy achieved is 97.66% with 31 features. However, the proposed model can generate a false alarm rate and cannot find the global optimal feature subset.

The authors of [25] presented a new approach for intrusion detection in SDN. The proposed system is a hybrid DL technique based on a convolutional neural network (CNN) and several ML algorithms, including Random Forest (RF), k-nearest neighbor (KNN), and SVM. Moreover, a new regularizer method, called SD-Reg, was introduced based on the weight matrix's standard deviation. The SD-Reg method addresses the problem of overfitting and improves the capability of network intrusion detection systems in detecting unseen intrusion events. The proposed technique outperformed the single DL models in all evaluation metrics. Moreover, the experimental results demonstrated that the CNN detects anomalies even with a few features. The combination of a CNN with the RF

algorithm achieved higher performance and can enhance intrusion detection accuracy. However, the complexity and computational requirements of the proposed model could be a disadvantage in practical applications despite its high accuracy in detecting intrusions in SDNs.

The authors of [26] combined Neighborhood Component Analysis (NCA) with ML and DL algorithms to categorize SDN traffic into normal or attack classes. NCA was utilized to select the most relevant features for prediction. Stochastic Gradient Descent (SGD) was also adopted to optimize the detection process's parameters. The experimental results show that Decision Trees (DTs) outperformed kNN, Artificial Neural Network (ANN), and SVM algorithms. The researchers also highlighted the importance of feature selection techniques in improving the precision of machine learning models for identifying attack traffic in SDN environments.

To identify DoS attacks, the authors of [27] developed a CNN model and evaluated its performance by comparing it with a recurrent neural network (RNN). The experiments were conducted for binary and multiclass classification using DoS datasets from the KDD Cup 1999 and CIC-IDS-2018. The authors transformed the symbolic data into numerical data to unify all data formats. The numerical data samples were converted into RGB and grayscale images; then, they trained their CNN model on both RGB and grayscale images. The authors concluded that the model performs better when trained on RGB images than on grayscale images in both binary and multiclass classifications. Furthermore, the comparison with the RNN model showed that the CNN model achieved higher accuracy for both KDD and CSE-CIC-IDS 2018 datasets.

The authors of [28] developed an intelligent DDoS attack detection model based on the Decision Tree algorithm and an improved Gini index feature selection technique. The UNSW-NB15 dataset was used to evaluate the model. Only 13 of the 45 security features were used, significantly reducing the dimensionality of the data and preventing overfitting problems. Additionally, the proposed approach performed better than baseline methods created using advanced algorithms like Random Forest and XGBoost.

Table 2 summarizes the general aspects of the related works described above, including the used models, the dataset, the best accuracy achieved, the number of features used, the feature selection approach, the balancing method employed, and the model output.

Based on the studied literature, it is evident that ML techniques are increasingly being utilized to identify DDoS attacks. The incorporation of data preprocessing and feature selection methods has enhanced various models. However, several approaches do not use recent datasets, limiting the examination of current DoS/DDoS attacks. Furthermore, while some models achieve high accuracy in classification, they suffer from the drawback of time complexity in the detection process. Thus, our study aims to develop an effective intrusion detection system based on MLP with high accuracy and minimal computational cost, and make it more energy-efficient to render the system well suited for deployment in IoT devices.

Table 2. General aspects of the cited research papers.

Paper	Model	Dataset	BestAcc %	Nb Features	Feature Selection Approach	Balancing Method	Multiclass
[9]	RBF-SVM	SDN-DDoS	99.40	-	PCA	-	No
[16]	GRU, LSTM, MLP, KNN, RF	CICDDoS2017	99.47 (LSTM) 99.97 (KNN)	49	removed highly correlated variables	-	Yes
		CICDDoS2019		50			
[19]	KNN, SVM, DT	Their own dataset	99.35 (DT)	25	minimum redundancy maximum relevance	-	Yes

Table 2. *Cont.*

Paper	Model	Dataset	BestAcc %	Nb Features	Feature Selection Approach	Balancing Method	Multiclass
[20]	DCAE	CICIDS2017, NSL-KDD, CIC-DDoS2019	97.58 96.08 92.45	-	-	-	No
[21]	Renyi joint entropy, ANN, XGB, SVM, KNN	SDN-DDoS	99.12 (ANN)	14	filter-based Fisher score, wrapper, ANOVA	-	No
[22]	CNN, GRU	CICIDS2017	99.7	39	-	-	No
[23]	CNN-GRU, SVM	NSL-KDD	98.45 (CNN-GRU)	-	PCA	-	No
[24]	MLP	NSL-KDD	97.66	31	sequential backward selection	-	No
[25]	CNN, RF, KNN, SVM,	CSE-CIC-IDS2018, UNSW-NB15	99.80 99.50	9	SD-Reg	SMOTE	Yes
[26]	ANN, DT, KNN, SVM	SDN-DDoS	100 (DT)	14	NCS	-	No
[27]	CNN	KDD CUP 1999, CSE-CIC-IDS2018	99.99	41 78	-	-	Yes
[28]	DT	UNSW-NB15	98	13	Gini index	-	Yes

3. Proposed Method

This study introduces a DL classifier, supported by feature selection and balancing methods called Eye-Net, for detecting and identifying DDoS attacks in IoT environments. The main objective is to produce a low-complexity and energy-efficient model that can accurately detect and classify DDoS attacks using ANOVA to obtain the most efficient and distinctive features, balancing the dataset by applying the SMOTE technique and reducing the computational cost in the deployment scenario using quantization. Eye-Net is evaluated for both binary and multiclass classifications. The process steps include data preprocessing and normalization stages, feature selection, data balancing for multiclass classification, DDoS attack classification and, quantization. An overall representation of the process's steps for Eye-Net is presented in Figure 1, and the pseudocode is presented in Algorithm 1.

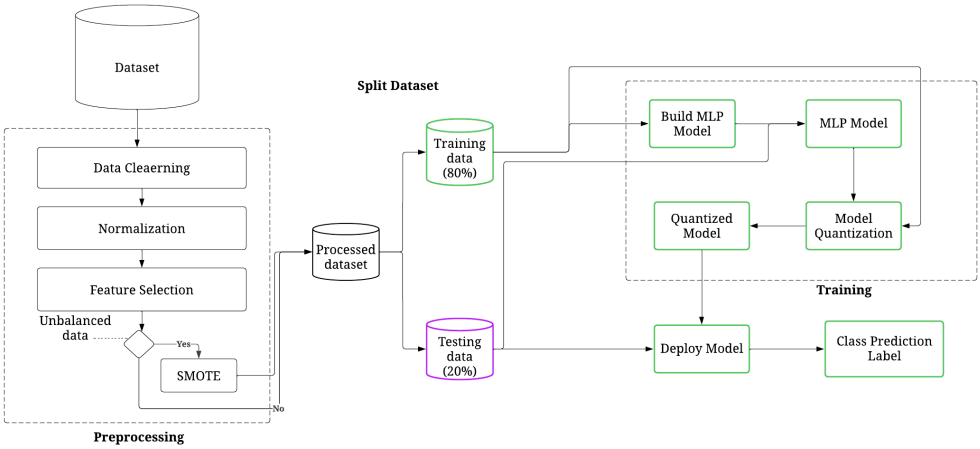


Figure 1. General workflow of Eye-Net.

Algorithm 1 Eye-Net

Input:
$X = \{x_1, x_2, x_3, ..., x_n\}$
$y = \{y_1, y_2, y_3, ..., y_n\}$
NF ▷ Number of selected features
Balanced = True ▷ True is the default value
MLP D_θ
Output:
Well learned parameters θ
Quantized Parameters θ_Q
$X = DataCleaning(X)$
$X = \frac{X - X_{min}}{X_{max} - X_{min}}$ ▷ Data Normalisation
$X = ANOVA(X, y, NF)$ ▷ Feature Selection
if $Balanced = True$ **then**
$X, y = SMOTE(X, y)$ ▷ Data Balancing
end if
$\theta \leftarrow$ initialize parameters
for each (x_i, y_i) in (X, y) **do**
$y' \leftarrow D_\theta(x_i, y_i)$
$\theta \leftarrow$ update parameters using AdamW
end for
$\theta_Q = Quantization(D_\theta(x, y))$

3.1. Preprocessing

Preprocessing is essential for creating an efficient model and minimizing computationally intensive processes.

3.1.1. Data Cleaning

Some cleaning methods are applied to enhance the data quality before the MLP model is trained. At this stage, irrelevant and noisy variables that fail to add distinctive characteristics to the classification process are eliminated, such as features containing only zeros and rows that include NaN or infinity values.

3.1.2. Feature Normalization

The data are rescaled using the MinMax normalization technique [29] according to Equation (1) to reduce the effects of the high-value variance between the features.

$$X_{norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \quad (1)$$

X_i, X_{min}, and X_{max} represent the feature's original, minimum, and maximum values, respectively. X_{norm} donates the normalized data ranging between 0 and 1.

3.1.3. Feature Selection

After preprocessing, the large amount of data dimensionality is reduced by applying the ANOVA technique to determine the most appropriate features in a dataset. ANOVA is a statistical technique that analyzes variables' variances to determine their differences. It is used to evaluate the statistical significance of each characteristic in the feature selection process. The target variable is compared to each feature to determine if there is a significant statistical relationship between them. ANOVA involves calculating the F-statistic and corresponding p-value for each feature. The F-statistic measures the variation between the means of different groups, while the p-value indicates the statistical significance of the result. The features that do not satisfy the statistical significance are removed, and those with the highest F-statistic and lowest p-value are chosen.

3.1.4. Data Balancing

Data balancing is achieved by applying the SMOTE technique, which generates synthetic examples of the minority class by interpolating new data points between existing minority class samples. It randomly selects a minority class sample and identifies its k-nearest neighbors. Synthetic samples are produced by calculating the difference between the feature vector under consideration and its nearest neighbor. This difference is then multiplied by a randomly generated value ranging from 0 to 1 and added to the original feature vector being examined.

3.2. Multilayer Perceptron (MLP) Classifier

This study proposes two Multilayer Perceptron (MLP) models to accurately classify DDoS attacks into binary and multiclass categories. The proposed models contain an input layer with m units and two hidden layers, each containing n neurons, where m and n are integer numbers. The Rectified Linear Unit (ReLU) activation function, defined in Equation (2), is utilized in the hidden layers. For the binary classification model, the output layer contains one neuron, and the sigmoid function described in Equation (3) is utilized as the activation function. In contrast, the multiclass classification model has an output layer consisting of seven neurons corresponding to the benign class and the six attack categories. The Softmax activation function, defined in Equation (4), is utilized as the activation function in the output layer of the multiclass classification model.

$$ReLU(x) = \max(0, x) \tag{2}$$

$$Sigmoid(x) = \frac{1}{1 + e^{-}x} \tag{3}$$

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=0}^{n} \exp(x_j)} \tag{4}$$

Both models are trained using the AdamW optimizer with a learning rate of 10^{-2} and a weight decay of 10^{-3}, which performs better than the Adam optimizer [30]. The binary cross-entropy cost represented in Equation (5) is used to train the binary classification model.

$$BJ(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{y}) + (1 - y^{(i)})\log(1 - \hat{y})] \tag{5}$$

where m represents the number of training examples, $y^{(i)}$ is the actual target value for the i-th training example, and $\hat{y}$ represents the predicted value.

The sparse categorical cross-entropy represented in Equation (6) is used to train the multiclass classification model.

$$CJ(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \tag{6}$$

m represents the number of training examples, C is the number of classes, y_{ij} is the actual target value for the i-th training example, and $\hat{y_{ij}}$ is the predicted value.

3.3. Quantization

Artificial Neural Networks have proven to be a reliable general-purpose tool to inject intelligence into many applications; however, they are computationally expensive due to the floating-point matrix–vector multiplication operation, which makes them impractical when they are deployed into IoT devices such as smartwatches, smart glasses, home appliances, drones, and robots because those devices require low-latency and energy-efficient solutions. Equation (7) represents the matrix–vector multiplication calculated in each layer in a neural network.

$$y = W \cdot x + b \tag{7}$$

W and b represent the learned weights and biases, respectively. They are usually stored in 16- or 32-bit precision.

Deploying neural Networks into IoT devices requires the design of sophisticated neural network accelerators to improve the inference time by performing as many calculations as possible in parallel. Neural network accelerators have two fundamental components: arithmetic processing units, $P_{n,m}$, and accumulators, A_n. Figure 2 represents a neural network accelerator that contains $n \times m$ arithmetic processing units and n accumulators.

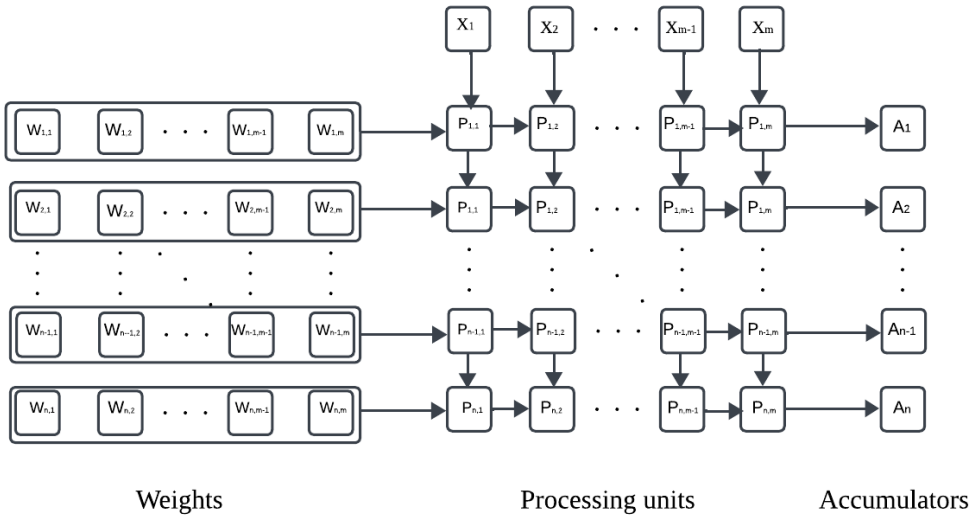


Figure 2. Matrix multiplication in neural network accelerator.

The calculation in the neural network accelerator starts by loading the accumulators A_n with the bias values b_n, followed by the weight values $W_{n,m}$ and the input values x_m. Next, it computes their product in the respective arithmetic processing units $P_{n,m}$. The results are added in the respective accumulator A_n according to Equation (8):

$$A_n = b_n + \sum_{i=1}^{i=m} W_{n,i} \times x_i \tag{8}$$

The operation represented in Equation (8) is also known as Multiply–Accumulate (MAC). MAC operations and data transfer consume a lot of energy in neural network inference due to the floating-point 32-bit representation of the weights and biases. A significant benefit can be achieved if the weights and biases represented on a lower-bit fixed point such as int8 lead to reducing the amount of data transfer, size, and energy consumption of MAC operation because the cost of arithmetic operations usually scales linearly to quadratically with the number of bits used. Furthermore, the fixed-point addition is more efficient than the floating-point counterpart [31].

In this study, a quantization-aware training technique is introduced to decrease the computational complexity of Eye-Net and make it suitable to operate on IoT devices by converting its weights and biases from 32-bit floating-point to 8-bit fixed-point representation. A floating-point vector can be expressed approximately by an integer vector divided by a non-zero integer scalar value, as shown in Equation (9):

$$X \approx \frac{X_{int}}{S_x} \tag{9}$$

The integer vector X_{int} can be obtained by rounding to the nearest results of the multiplication of the floating-point vector X by the integer scalar S_x, as shown in Equation (10):

$$X_{int} = \lfloor S_x \cdot X \rceil \tag{10}$$

where $\lfloor \cdot \rceil$ is the round-to-nearest operator. The values of the vector X_{int} lie in the range $[-2^{b-1}, 2^{b-1} - 1]$, where b represents the number of bits used to represent the integer values. By quantizing the weights and biases according to Equations (9) and (10), the accumulation equation can be written as follows:

$$\begin{aligned} A_n &\approx \frac{b_n^{int} + \sum_{i=1}^{i=m} W_{n,i}^{int} \times x_i}{S} \\ &\approx \frac{\lfloor S \times b_n \rceil + \sum_{i=1}^{i=m} \lfloor S \times W_{n,i} \rceil \times x_i}{S} \end{aligned} \tag{11}$$

Equation (11) shows two significant problems. The first problem is the division operation by the scalar S after each accumulation, which scales linearly with the number of neurons in a neural network and consequently affects the hardware efficiency and energy consumption. For example, a neural network with 7 thousand neurons requires 7 thousand divisions by the scalar S. To solve this issue, the choice of the scalar S was restricted to the power of two values, $S = 2^k$, where k is an integer number. Dividing by the power of two values is just a simple bit-shifting operation, making it energy-efficient. The accumulation equation can be written as shown in Equation (12):

$$A_n \approx \frac{\lfloor 2^k \times b_n \rceil + \sum_{i=1}^{i=m} \lfloor 2^k \times W_{n,i} \rceil \times x_i}{2^k} \tag{12}$$

On the other hand, the round-to-nearest operation employed in quantization introduces noise to the accumulated values, significantly degrading the neural network's performance. To solve this issue, the weights and biases in each layer should be quantized separately layer by layer, starting with the first layer; then, its weights and biases should be kept untrainable and one should retrain the neural network to update the weights and biases of the other layers to correct the error caused by the quantization operation in that layer. Those steps should be applied to each layer in the neural network until all the neural network layers are quantized as represented in the Algorithm 2.

Algorithm 2 Quantization

Input: D, θ, i, X, y, $l_{idx} \leftarrow 1$
Output: θ_{int8}
function QUANTIZE_WEIGHTS(D, θ, l_{idx}, i)
  $temp_\theta \leftarrow \theta$
  $q \leftarrow \frac{\lfloor 2^i \cdot \theta[l_{idx}] \rceil}{2^i}$
  $temp_\theta[l_{idx}] \leftarrow q$
  D.layers[l_{idx}].trainable $\leftarrow$ False
  $l_{idx} \leftarrow l_{idx} + 1$
  return $temp_\theta, l_{idx}$
end function
$\theta_{int8}, l_{idx} \leftarrow$ QUANTIZE_WEIGHTS(D, θ, l_{idx}, i)
for _ in len(D.layers) **do**
  $\theta_{int8} \leftarrow$ train(D, θ_{int8}, X, y)
  $\theta_{int8}, l_{idx} \leftarrow$ QUANTIZE_WEIGHTS(D, θ_{int8}, l_{idx}, i)
end for

D, θ, (X,y), l_{idx}, and θ_{int8} represent the Tensorflow model, the model's weights and biases, the training data, the layer index, and the quantized model weights and biases, respectively.

3.4. Complexity Analysis

Eye-Net's complexity is calculated by the sum of every stage's computational complexity, including data cleaning, normalization, feature selection, balancing, classification, and quantization. Let n be the size of the dataset and f the number of features. The data cleaning step's complexity depends on the dataset's size. In the worst case, it can be approximately equal to $O(n)$, where each sample requires cleaning. The computational complexity of the data normalization step is also $O(n)$. On the other hand, the feature selection step employs ANOVA, which has a time complexity of $O(nl)$. The number of features in this study is fixed; hence, this step's time complexity is $O(n)$. In the data balancing step, the time complexity of SMOTE is $O(mlog_2m)$, where m represents the number of samples. SMOTE identifies the k-nearest neighbor samples and generates new synthetic samples for each minority sample, resulting in time complexities of $O(m)$ and $O(1)$, respectively. The computational complexity of the classifier is linear, primarily determined by the fixed sequence of vector–matrix multiplication in the MLP classifier, which operates with a complexity of $O(p)$, where p represents the number of rows in the matrix. The quantization's complexity is equal to the complexity of MLP multiplied by the number of layers l. As the model's number of layers is fixed, the complexity of this phase is $O(p)$. During the training phase of Eye-Net, its complexity is $O(p \log_2 p)$. However, it becomes linear with $O(p)$ complexity during deployment, wherein only the classifier is utilized.

4. Experiments and Results

4.1. Dataset

The CICDDoS2019 dataset developed by [32] was used in this study. It was created using the open-source CICFlowMeter, which extracted 86 variables from the network traffic flow. It comprises 12 different types of modern DDoS attacks that can be executed using TCP/UDP application layer protocols. These attacks are mainly divided into two categories: reflection and exploitation (as shown in Figure 3. Reflection attacks are DDoS methods where the attacker tricks a server into sending a large volume of traffic to a target by spoofing the target's IP address in the requests, while exploitation attacks involve using vulnerabilities in software or systems to disrupt services and overwhelm the target with malicious traffic. Both hide the attacker's identity and flood the victim's resources with response packets by sending packets to reflector servers using the victim's IP address as the source IP address.

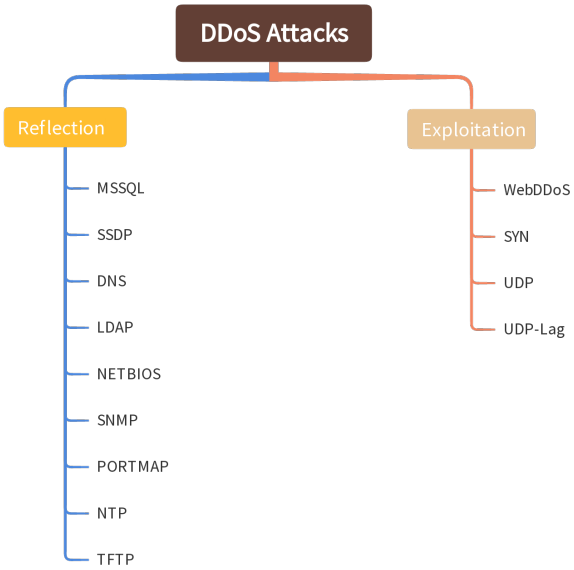


Figure 3. Classification of DDoS attack types.

4.2. Evaluation Metrics

Standard metrics, such as accuracy (Equation (13)), recall (Equation (14)), precision (Equation (15)), and F1 Score (Equation (16)), are used to evaluate the performance of the proposed model. The accuracy indicates the model's rate of correct predictions. Recall represents true positives successfully predicted by the model, whereas precision represents positives correctly predicted by the model. The F1 Score shows the stability between recall and Sensitivity. A Confusion Matrix is also used to determine the model's learning requirements and to understand how it makes correct and wrong predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{16}$$

TP, TN, FP, and FN represent True Positive, True Negative, False Positive, and False Negative, respectively.

4.3. Results

To evaluate Eye-Net, the CICDDoS2019 dataset was used in the experiments. At first, we performed data cleaning by deleting the rows that contained 'NaN' or 'infinity' values. Eight features (Flow ID, SourceIP, SourcePort, DestinationIP, DestinationPort, Protocol, Timestamp, SimillarHTTP) that do not contribute to the training and nine features (Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Bytes/Bulk Avg, Fwd Bulk Rate Avg, Fwd Packet/Bulk Avg, Bwd Bytes/Bulk Avg, Bwd Packet/Bulk Avg, Bwd Bulk Rate Avg) with 'infinity' values were removed from the dataset because they contain only the value of '0'. Then, feature normalization was applied to the remaining 69 features to speed up the training process. The ANOVA feature selection method was applied multiple times to the normalized data with different numbers of features {10, 15, 20, 25, 30} to determine the optimal number of features to reduce the complexity of the models and preserve their performance. Table 3 represents the selected features in the feature selection step. Multiple datasets were created: five datasets for the binary classification and another five for the multiclass classification. In the binary classification datasets, the class label "BENIGN" was designated as "0", while other attack types were designated as "1". The attacks were labeled from "1" to "6" for the multiclass classification datasets. The NetBios and Portmap attacks were merged due to the high similarity between their feature values. SMOTE was applied to the minority classes in the multiclass classification datasets. To train our models, all datasets were partitioned into 80% for training and 20% for testing purposes.

Table 3. The selected features for Eye-Net.

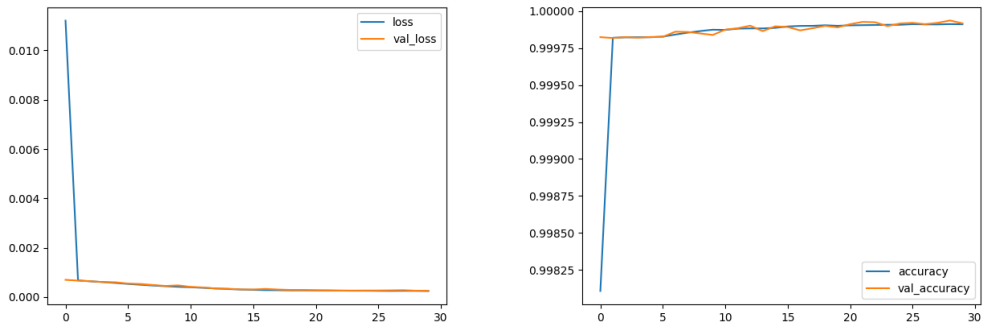
Feature	Description	Feature	Description	Feature	Description	Feature	Description
Total Length of Fwd Packets	Total packets in the forward direction	Fwd Packet Length Max	Maximum size of packet in forward direction	CWE Flag Count	Number of packets with CWE	Down/Up Ratio	Download and upload ratio
Fwd Packet Length Min	Minimum size of packet in forward direction	Fwd Packet Length Mean	Mean size of packet in forward direction	Average Packet Size	Average size of packet	Avg Fwd Segment Size	Average forward direction segment size

Table 3. *Cont.*

Feature	Description	Feature	Description	Feature	Description	Feature	Description
Fwd Packet Length Std	Standard deviation size of packet in forward direction	Bwd Packet Length Max	Maximum size of packet in backward direction	Avg Bwd Segment Size	Average backward direction segment size	Subflow Fwd Bytes	The average number of bytes in a subflow in the forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction	Bwd Packet Length Mean	Mean size of packet in backward direction	Init Win bytes forward	Initial Window Size in Bytes Forward	Inbound	Direction traffic moves between networks
Bwd Packet Length Std	Standard deviation size of packet in backward direction	Flow Bytes/s	Number of flow bytes per second	ACK Flag Count	Number of packets with ACK	URG Flag Count	Number of packets with URG
Flow Packets/s	Number of flow packets per second	Fwd PSH Flags	Number of times the PSH flag was set in packets	Packet Length Variance	Packet Length Variance	RST Flag Count	Number of packets with RST
Fwd Packets/s	Number of forward packets per second	Bwd Packets/s	Number of backward packets per second	Packet Length Mean	Mean size of packet length	Packet Length Std	Standard deviation size of packet length
Min Packet Length	Minimum size of packet length	Max Packet Length	Maximum size of packet length				

4.3.1. Results of the Binary Classification

Several models with various architectures were trained on the five datasets of the binary classification. The names of models were coded as BF-N, where B refers to binary, F refers to the number of input features, and N refers to the number of neurons in each hidden layer. Each model was trained for 30 epochs. Table 4 shows that all models reach excellent performance, ranging from 0.02% to 0.09% in error rate and from 99.97% to 99.99% in terms of accuracy, recall, precision, and F1 Score. The best model, B30-64, achieved nearly 100% accuracy and an error rate close to 0%, as demonstrated in Figure 4. To provide detailed information regarding the model's performance in the classification task, a normalized confusion matrix is also presented in Figure 5. The normalized confusion matrix indicated that B30-64 can accurately classify the flow packets during the testing phase.



(**a**) Error of the model (**b**) Accuracy of the model

Figure 4. The performance of the B30-64 model.

Table 4 also shows that the B30-32 and B25-64 models exhibit similar performance to B30-64 with a slight loss in error rate. B30-32 and B25-64 use fewer parameters than B30-64, especially B30-32, which uses one-third of the parameters used in B30-64, making B30-32 much better than B30-64 in terms of energy efficiency. The quantization was applied to the B30-64 model using scalar factor $S = 2^4$. The performance of B30-64 was evaluated after quantizing the weights and biases of B30-64 to INT8. The quantized version of B30-64 showed the same performance as its counterpart. The quantized model B30-64Q was also compared with recent state-of-the-art DDoS attack detection systems regarding the number

of features used, error rate, precision, recall, F1 Score, and accuracy. Table 5 shows that B30-64 outperforms the state-of-the-art DDoS attack detection systems in all performance metrics, reaching 99.99% accuracy. At the same time, the error rate, precision, recall, and F1 score all remain very competitive at 0.02%, 99.99%, 99.99%, and 99.99%, respectively. The authors of [13,33–35] used large input feature vectors as well as more extensive and complex models compared to ours. However, they still cannot outperform our model.

Table 4. Performance of the different MLP models in binary classification.

Model	Loss	Accuracy	Val-Loss	Val-Accuracy	Precision	Recall	F1 Score	N Parms
B10-16	0.0011	0.9997	0.0009	0.9997		0.9997	0.9998	**465**
B10-32	0.0009	0.9997	0.0008	0.9997		0.9997	0.9998	1441
B10-64	0.0008	0.9997	0.0007	0.9998		0.9998	0.9998	4929
B15-16	0.0008	0.9998	0.0008	0.9998		0.9998	0.9998	545
B15-32	0.0005	0.9998	0.0005	0.9998		0.9998	0.9998	1601
B15-64	0.0007	0.9998	0.0007	0.9998		0.9998	0.9998	5249
B20-16	0.0007	0.9998	0.0007	0.9998		0.9998	0.9998	625
B20-32	0.0006	0.9998	0.0006	0.9998		0.9998	0.9998	1761
B20-64	0.0006	0.9998	0.0006	0.9998	**0.9999**	0.9999	0.9999	5569
B25-16	0.0006	0.9998	0.0006	0.9998		0.9999	0.9999	705
B25-32	0.0005	0.9998	0.0005	0.9998		0.9999	0.9999	1921
B25-64	0.0003	0.9999	0.0003	0.9999		0.9999	0.9999	5889
B30-16	0.0005	0.9998	0.0005	0.9998		0.9999	0.9999	785
B30-32	0.0003	0.9999	0.0003	0.9999		0.9999	0.9999	2081
B30-64	**0.0003**	**0.9999**	**0.0002**	**0.9999**		**0.9999**	**0.9999**	6209
B30-32-60	0.0003	0.9999	0.0003	0.9999		0.9999	0.9999	2081
B25-64-60	0.0003	0.9999	0.0003	0.9998		0.9999	0.9999	5889

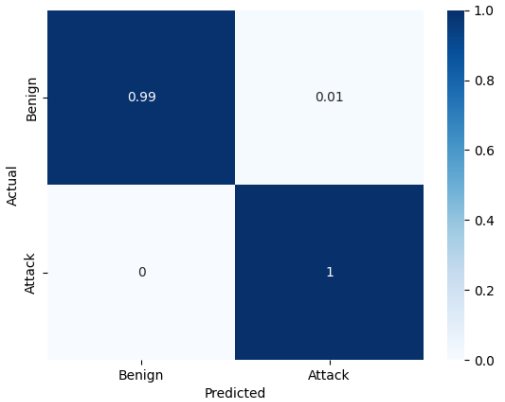


Figure 5. Normalized confusion matrix of B30-64.

Table 5. Comparison of B30-64 with state-of-the-art DDOS attack detection systems.

Paper	Method	Number of Features	Loss	Precision	Recall	F1 Score	Accuracy
[13]	RNN-Autoencoder	77	<0.0025	0.9950	0.99	0.99	0.99
[33]	DNN	69	>0.10	0.9999	0.9998	0.9998	0.9997
[34]	EDSA, DNN using autoencoder	80	<0.01	0.91	0.981	0.9441	0.98
[35]	Bidirectional LSTM-GMM	80	-	0.895	0.953	0.923	0.942
Eye-Net	MLP (B30-64Q)	**30**	**0.0002**	**0.9999**	**0.9999**	**0.9999**	**0.9999**

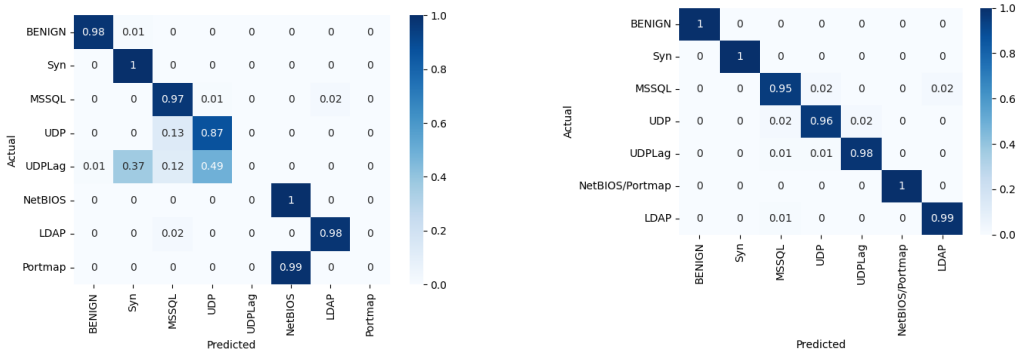
4.3.2. Results of the Multiclass Classification

Several models with various architectures were trained on the five datasets of multiclass classification. The names of models were coded as MF-N, where M refers to multiclass, F refers to the number of input features, and N is the number of neurons in each hidden layer. Each model was trained for 30 epochs. Before training and evaluating the different architectures of the MLP classifier, the M30-64 model was selected to illustrate the importance of the balancing step. Two scenarios were considered: the first consisted of training the model without balancing the dataset, while the second involved balancing the dataset using the SMOTE technique and unifying Portmap and NetBIOS classes into one label.

In the first scenario, M30-64 was trained to classify the eight labels presented in the multiclass dataset containing 30 features without balancing. The M30-64 model performed poorly in all evaluation metrics, as shown in Table 6. Figure 6a represents the normalized confusion matrix during testing. It is observed that the model failed to classify flow packets related to UD-PLag and Portmap DDoS attacks. The misclassification of UDPLag is due to the insufficient number of samples available in the imbalanced dataset. On the other hand, the model identified flow packets of Portmap attacks as NetBIOS attacks because of the high similarity in the feature space between these two attacks, as shown in Figure 7.

Table 6. The performance of the M30-64 model with and without SMOTE.

Model	Loss	Accuracy	Val-Loss	Val-Accuracy	Precision	Recall	F1 Score
M30-64 without SMOTE	0.1089	0.9691	0.1100	0.9559	0.7526	0.7257	0.7228
M30-64 with SMOTE	0.0743	0.9775	0.0709	0.9807	0.9772	0.9822	0.9796



(**a**) (**b**)

Figure 6. Normalized confusion matrix for multiclass classification of M30-64. (**a**) Normalized Confusion matrix for multiclass classification without oversampling of M30-64. (**b**) Normalized confusion matrix for multiclass classification with oversampling of M30-64.

Consequently, in the second scenario, the SMOTE balancing technique was applied to the dataset used in the first scenario to avoid any implicit biases. Additionally, we decided to unify NetBIOS and Portmap attacks into the same class. The model was then retrained.

Table 6 shows that the performance of M30-64 significantly improved compared to the first scenario. M30-64 achieved 7.09% and 98.07% in terms of error rate and accuracy, respectively, as demonstrated in Figure 8. Moreover, the normalized confusion matrix in Figure 6b demonstrates that M30-64 can accurately classify the flow packets into their appropriate classes.

Various architectures of the MLP classifier were trained and evaluated to identify the optimal configuration that produces the best results. Table 7 highlights that M30-64 performs better than the other models under different metrics. On the other hand, the quantization was applied to the M30-64 model using scalar factor $S = 2^3$. Table 8 shows that the performance of the quantized version of M30-64 slightly decreased compared

to its counterpart with the floating-point weights and biases. The normalized confusion matrix in Figure 9 indicates that the quantized M30-64Q model can accurately classify the flow packets into their appropriate classes except for the UDP class, where the accuracy decreased by 13% compared to that of M30-64 for the same class.

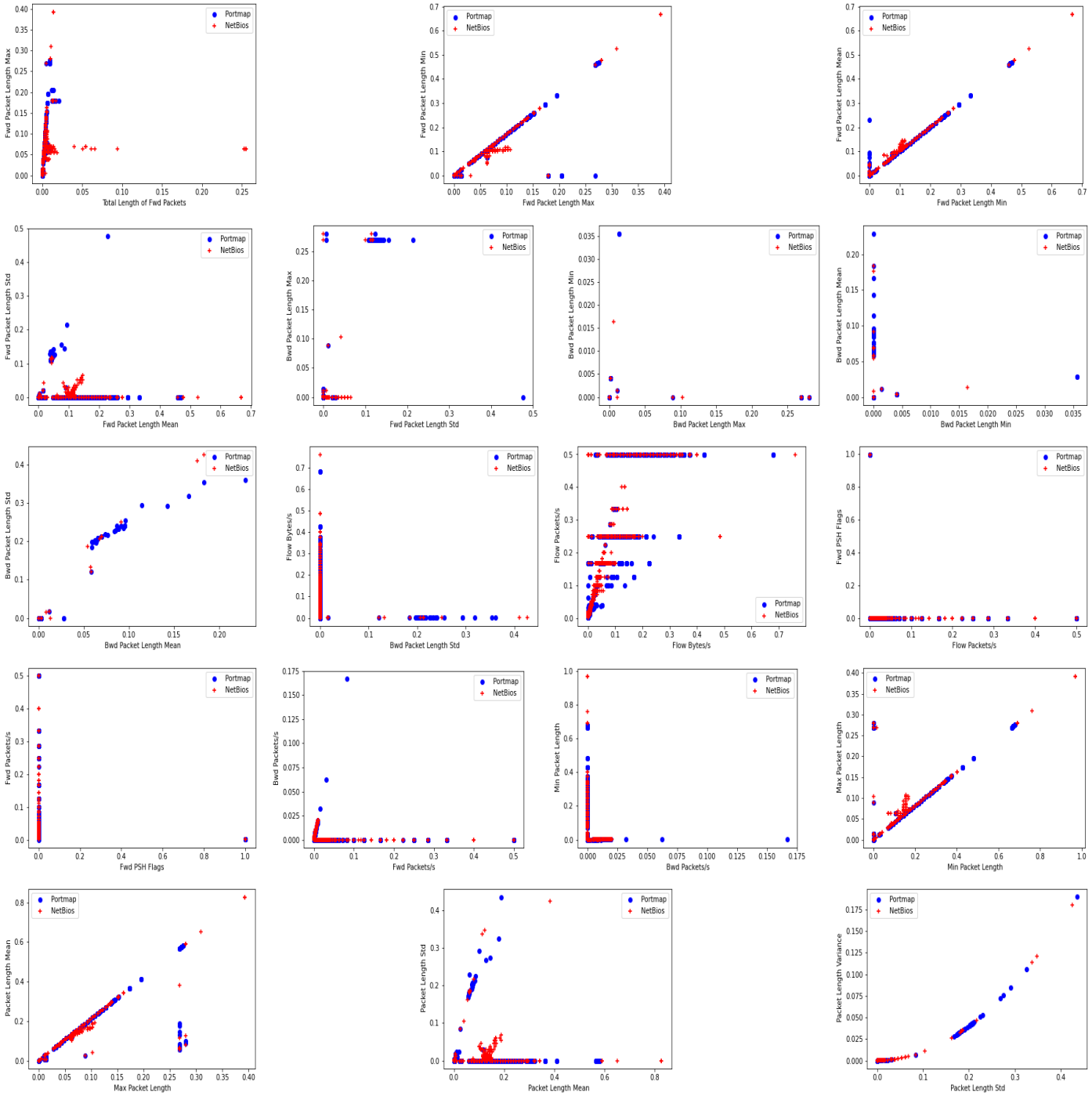
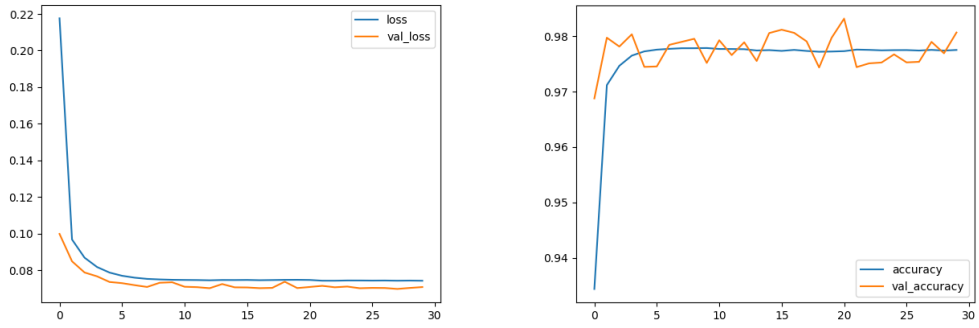


Figure 7. The similarity between Portmap and Netbios features.



(**a**) Error of the model (**b**) Accuracy of the model

Figure 8. The performance of the M30-64 model.

Table 7. Performance of the different MLP models in multiclass classification.

Model	Loss	Accuracy	Val-Loss	Val-Accuracy	Precision	Recall	F1 Score	N Parms
M10-16	0.1816	0.9449	0.2232	0.8842	0.7717	0.8152	0.7922	**567**
M10-32	0.1318	0.9591	0.1257	0.9605	0.9590	0.9640	0.9612	1639
M10-64	0.1131	0.9642	0.1117	0.9705	0.9696	0.9710	0.9701	5319
M15-16	0.1403	0.9593	0.1318	0.9630	0.9627	0.9656	0.9638	647
M15-32	0.1064	0.9677	0.0986	0.9693	0.9667	0.9717	0.9691	1799
M15-64	0.0900	0.9728	0.0838	0.9727	0.9667	0.9717	0.9691	5639
M20-16	0.1359	0.9588	0.1539	0.9620	0.9667	0.9717	0.9691	727
M20-32	0.0994	0.9694	0.0983	0.9756	0.9735	0.9760	0.9747	1959
M20-64	0.0836	0.9747	0.0778	0.9785	0.9753	0.9798	0.9774	5959
M25-16	0.1241	0.9626	0.1254	0.9662	0.9680	0.9559	0.9613	807
M25-32	0.1007	0.9696	0.0933	0.9721	0.9694	0.9736	0.9714	2119
M25-64	0.0829	0.9751	0.0785	0.9802	0.9774	0.9807	0.9789	6279
M30-16	0.1115	0.9646	0.1238	0.9621	0.9611	0.9653	0.9627	887
M30-32	0.0872	0.9724	0.0901	0.9724	0.9702	0.9734	0.9717	2279
M30-64	**0.0743**	**0.9775**	**0.0709**	**0.9807**	**0.9772**	**0.9822**	**0.9796**	6599

Table 8. Comparison between M30-64 and its quantized version, M30-64Q.

Model	Loss	Accuracy	Precision	Recall	F1 Score
M30-64	0.0709	0.9807	0.9772	0.9822	0.9796
M30-64Q	0.0878	0.9647	0.9673	0.9637	0.9643

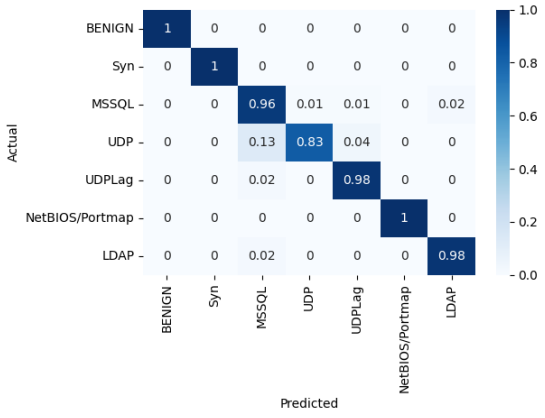


Figure 9. Normalized confusion matrix after quantization of M30-64Q.

The results of the M30-64Q model were compared with recent state-of-the-art DDoS attack detection systems in terms of the number of used features and evaluation metrics. Table 9 shows that Eye-Net outperforms the state-of-the-art DDoS attack detection systems for different evaluation metrics, achieving 96.47% accuracy. The error rate, precision, recall, and F1 score remain highly competitive at 8.78%, 96.73%, 96.73%, and 96.43%, respectively, except for the approaches proposed in [36,37], which are slightly superior to M30-64Q. However, our model is much better regarding energy efficiency, memory consumption, and inference time because the competitor method requires a feature extraction step to classify flow packets. The weights and biases are represented in 32-bit floating-point precision, making the method unsuitable for deployment in real-world applications, especially in IoT environments.

Table 9. Comparison of M30-64 and M30-64Q with state-of-the-art DDOS attack detection systems.

Paper	Method	Number of Features	Loss	Precision	Recall	F1 Score	Accuracy
[33]	DNN	69	<0.12	0.8049	0.9515	0.8721	0.9457
[38]	DNN	72	-	0.9421	0.9403	0.9412	0.9421
[39]	CNN	86	-	0.90	0.90	0.90	0.9590
[36]	AE-MLP	78	-	0.9791	0.9848	0.9818	0.9834
[37]	MLP-CNN	-	-	**0.999**	**0.9998**	**0.9994**	**0.9995**
Eye-Net	MLP (M30-64)	**30**	**0.0709**	0.9772	0.9822	0.9796	0.9807
Eye-Net	MLP (M30-64Q)	**30**	0.0878	0.9673	0.9637	0.9643	0.9647

5. Limitations

Eye-Net has two drawbacks despite its considerable potential and excellent performance. The first limitation is that Eye-Net only classifies six types of DDoS attacks in the multiclass classification. This restriction means that it may not adequately detect or categorize emerging or less common types of DDoS attacks. As new attack vectors are continually developed, the system's ability to stay updated and adapt to these changes becomes crucial. Currently, Eye-Net's scope does not extend beyond the predefined six types. The second disadvantage is not considering the quantization of the input vector to a lower-bit precision, such as utilizing an INT8 fixed-point representation. This omission results in higher memory usage and more energy consumption than a quantized input vector.

6. Conclusions and Future Work

Distributed Denial of Service (DDoS) attacks present a significant danger to the availability and integrity of online services. A deep learning-based DDoS detection system called Eye-Net has been developed to address this issue. It employs an optimized and low-complexity Multilayer Perceptron (MLP) architecture to accurately identify and classify DDoS attacks. The ANOVA algorithm is applied to select distinctive features, and Synthetic Minority Oversampling (SMOTE) is performed to generate samples for minority classes. Eye-Net also employs quantization to the MLP classifier to represent its weights and biases in a lower-bit fixed-point precision, making it suitable for deployment in edge devices. Experimental results on the CICDDoS2019 dataset show that Eye-Net achieved high accuracy and provided the best trade-off between performance and energy efficiency compared to the state-of-the-art DDoS detection methods in binary and multiclass classifications.

While the current study highlights the accuracy and robustness of the Eye-Net system for DDoS attack detection, future work will focus on several key areas to further optimize and expand its capabilities. We plan to conduct a comprehensive analysis of the time, energy, and resource consumption associated with deploying Eye-Net, particularly within the context of IoT environments. This analysis will include measurements of inference time, energy usage, and memory footprint on typical IoT hardware. Additionally, we aim to enhance the multiclass classification performance of Eye-Net and expand its detection capabilities to include a broader range of DDoS attack types.

To improve the model's operational efficiency, we will explore quantizing the input vector to a lower-bit representation, reducing energy consumption during data transfer, and making Eye-Net more suitable for resource-constrained environments. Furthermore, we will consider incorporating additional features and information sources to further boost Eye-Net's accuracy and effectiveness. These enhancements will ensure that Eye-Net not only maintains its high detection performance but also becomes more adaptable and efficient for real-world IoT applications.

Author Contributions: R.K.: Conceptualization, Methodology, Formal analysis, Investigation, Implementation, Validation, Writing—original draft. I.G.: Conceptualization, Investigation, Supervision, Writing, Reviewing and editing. M.A.F.: Reading, Writing, Reviewing and editing, Validation. All authors have read and agreed to the published version of the manuscript.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement: Data will be made available on request.

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

References

1. Polat, H.; Türkoglu, M.; Polat, O.; Sengür, A. A novel approach for accurate detection of the DDoS attacks in SDN-based SCADA systems based on deep recurrent neural networks. *Expert Syst. Appl.* **2022**, *197*, 116748 . [CrossRef]
2. Halladay, J.; Cullen, D.; Briner, N.; Warren, J.; Fye, K.; Basnet, R.; Bergen, J.; Doleck, T. Detection and Characterization of DDoS Attacks Using Time-Based Features. *IEEE Access* **2022**, *10*, 49794–49807. [CrossRef]
3. Choi, J.; Choi, C.; Ko, B.; Kim, P. A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment. *Soft Comput.* **2014**, *18*, 1697–1703. [CrossRef]
4. Quezada, V.; Astudillo-Salinas, F.; Tello-Oquendo, L.; Bernal, P. Real-time bot infection detection system using DNS fingerprinting and machine-learning. *Comput. Netw.* **2023**, *228*, 109725. [CrossRef]
5. Mittal, M.; Kumar, K.; Behal, S. Deep learning approaches for detecting DDoS attacks: A systematic review. *Soft Comput.* **2023**, *27*, 13039–13075. [CrossRef]
6. Mehmood, A.; Mukherjee, M.; Ahmed, S.H.; Song, H.; Malik, K.M. NBC-MAIDS: Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks. *J. Supercomput.* **2018**, *74*, 5156–5170. [CrossRef]
7. Khare, M.; Oak, R. Real-Time distributed denial-of-service (DDoS) attack detection using decision trees for server performance maintenance. In *Performance Management of Integrated Systems and Its Applications in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–9. Available online: https://link.springer.com/chapter/10.1007/978-981-13-8253-6_1 (accessed on 1 July 2024).
8. Chen, L.; Zhang, Y.; Zhao, Q.; Geng, G.; Yan, Z. Detection of DNS DDoS Attacks with Random Forest Algorithm on Spark. *Procedia Comput. Sci.* **2018**, *134*, 310–315. [CrossRef]
9. Anyanwu, G.O.; Nwakanma, C.I.; Lee, J.M.; Kim, D.S. RBF-SVM kernel-based model for detecting DDoS attacks in SDN integrated vehicular network. *Ad Hoc Netw.* **2023**, *140*, 103026. [CrossRef]
10. Zhang, L.; Jiang, S.P.; Shen, X.; Gupta, B.B.; Tian, Z. PWG-IDS: An Intrusion Detection Model for Solving Class Imbalance in IIoT Networks Using Generative Adversarial Networks. *arXiv* **2021**, arXiv:abs/2110.03445.
11. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
12. Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407.
13. Elsayed, M.S.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Ddosnet: A deep-learning model for detecting network attacks. In Proceedings of the 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 31 August–3 September 2020; IEEE: Washington, DC, USA, 2020; pp. 391–396.
14. Huang, S.; Lei, K. IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Netw.* **2020**, *105*, 102177. [CrossRef]
15. Nie, L.; Wu, Y.; Wang, X.; Guo, L.; Wang, G.; Gao, X.; Li, S. Intrusion Detection for Secure Social Internet of Things Based on Collaborative Edge Computing: A Generative Adversarial Network-Based Approach. *IEEE Trans. Comput. Soc. Syst.* **2022**, *9*, 134–145. [CrossRef]
16. Yungaicela-Naula, N.M.; Vargas-Rosales, C.; Perez-Diaz, J.A. SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning. *IEEE Access* **2021**, *9*, 108495–108512. [CrossRef]
17. Elssied, N.O.F.; Ibrahim, O.; Osman, A.H. A novel feature selection based on one-way anova f-test for e-mail spam classification. *Res. J. Appl. Sci. Eng. Technol.* **2014**, *7*, 625–638. [CrossRef]
18. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]

19. Türkoğlu, M.; Polat, H.; Koçak, C.; Polat, O. Recognition of DDoS attacks on SD-VANET based on combination of hyperparameter optimization and feature selection. *Expert Syst. Appl.* **2022**, *203*, 117500. [CrossRef]
20. Aktar, S.; Nur, A.Y. Towards DDoS attack detection using deep learning approach. *Comput. Secur.* **2023**, *129*, 103251. [CrossRef]
21. Wang, Y.; Wang, X.; Ariffin, M.M.; Abolfathi, M.; Alqhatani, A.; Almutairi, L. Attack detection analysis in software-defined networks using various machine learning method. *Comput. Electr. Eng.* **2023**, *108*, 108655. [CrossRef]
22. Diaba, S.Y.; Elmusrati, M. Proposed algorithm for smart grid DDoS detection based on deep learning. *Neural Netw.* **2023**, *159*, 175–184. [CrossRef]
23. Ahmad, I.; Wan, Z.; Ahmad, A. A big data analytics for DDOS attack detection using optimized ensemble framework in Internet of Things. *Internet Things* **2023**, *23*, 100825. [CrossRef]
24. Wang, M.; Lu, Y.; Qin, J. A dynamic MLP-based DDoS attack detection method using feature selection and feedback. *Comput. Secur.* **2020**, *88*, 101645. [CrossRef]
25. ElSayed, M.S.; Le-Khac, N.A.; Albahar, M.A.; Jurcut, A. A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *J. Netw. Comput. Appl.* **2021**, *191*, 103160. [CrossRef]
26. Tonkal, Ö.; Polat, H.; Başaran, E.; Cömert, Z.; Kocaoğlu, R. Machine learning approach equipped with neighbourhood component analysis for DDoS attack detection in software-defined networking. *Electronics* **2021**, *10*, 1227. [CrossRef]
27. Kim, J.; Kim, J.; Kim, H.; Shim, M.; Choi, E. CNN-based network intrusion detection against denial-of-service attacks. *Electronics* **2020**, *9*, 916. [CrossRef]
28. Bouke, M.A.; Abdullah, A.; ALshatebi, S.H.; Abdullah, M.T.; El Atigh, H. An intelligent DDoS attack detection tree-based model using Gini index feature selection method. *Microprocess. Microsyst.* **2023**, *98*, 104823. [CrossRef]
29. Patro, S.; Sahu, K.K. Normalization: A preprocessing stage. *arXiv* **2015**, arXiv:1503.06462.
30. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
31. Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In Proceedings of the 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, USA, 9–13 February 2014; IEEE: Washington, DC, USA, 2014; pp. 10–14.
32. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; IEEE: Washington, DC, USA, 2019; pp. 1–8.
33. Cil, A.E.; Yildiz, K.; Buldu, A. Detection of DDoS attacks with feed forward based deep neural network model. *Expert Syst. Appl.* **2021**, *169*, 114520. [CrossRef]
34. Sindian, S.; Samer, S. An enhanced deep autoencoder-based approach for DDoS attack detection. *Wseas Trans. Syst. Control* **2020**, *15*, 716–725. [CrossRef]
35. Shieh, C.S.; Lin, W.W.; Nguyen, T.T.; Chen, C.H.; Horng, M.F.; Miu, D. Detection of unknown ddos attacks with deep learning and gaussian mixture model. *Appl. Sci.* **2021**, *11*, 5213. [CrossRef]
36. Wei, Y.; Jang-Jaccard, J.; Sabrina, F.; Singh, A.; Xu, W.; Camtepe, S. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access* **2021**, *9*, 146810–146821. [CrossRef]
37. Setitra, M.A.; Fan, M.; Agbley, B.L.Y.; Bensalem, Z.E.A. Optimized MLP-CNN Model to Enhance Detecting DDoS Attacks in SDN Environment. *Network* **2023**, *3*, 538–562. [CrossRef]
38. Chartuni, A.; Márquez, J. Multi-Classifier of DDoS Attacks in Computer Networks Built on Neural Networks. *Appl. Sci.* **2021**, *11*, 10609. [CrossRef]
39. Ferrag, M.A.; Shu, L.; Djallel, H.; Choo, K.K.R. Deep learning-based intrusion detection for distributed denial of service attack in Agriculture 4.0. *Electronics* **2021**, *10*, 1257. [CrossRef]