electronics

MDPI

Article

Network Attack Classification with a Shallow Neural Network for Internet and Internet of Things (IoT) Traffic

Jörg Ehmer [1,*], Yvon Savaria [1], Bertrand Granado [2], Jean-Pierre David [1] and Julien Denoulet [2]

[1] Electrical Engineering Department, Ecole Polytechnique, Montréal, QC H3T 1J4, Canada; yvon.savaria@polymtl.ca (Y.S.); jean-pierre.david@polymtl.ca (J.-P.D.)
[2] Campus Pierre et Marie Curie, Sorbonne Université, CNRS, LIP6, F-75005 Paris, France; bertrand.granado@sorbonne-universite.fr (B.G.); julien.denoulet@sorbonne-universite.fr (J.D.)
* Correspondence: jorg.ehmer@polymtl.ca

Abstract: In recent years, there has been a tremendous increase in the use of connected devices as part of the so-called Internet of Things (IoT), both in private spaces and the industry. Integrated distributed systems have shown many benefits compared to isolated devices. However, exposing industrial infrastructure to the global Internet also generates security challenges that need to be addressed to benefit from tighter systems integration and reduced reaction times. Machine learning algorithms have demonstrated their capacity to detect sophisticated cyber attack patterns. However, they often consume significant amounts of memory, computing resources, and scarce energy. Furthermore, their training relies on the availability of datasets that accurately represent real-world data traffic subject to cyber attacks. Network attacks are relatively rare events, as is reflected in the distribution of typical training datasets. Such imbalanced datasets can bias the training of a neural network and prevent it from successfully detecting underrepresented attack samples, generally known as the problem of imbalanced learning. This paper presents a shallow neural network comprising only 110 ReLU-activated artificial neurons capable of detecting representative attacks observed on a communication network. To enable the training of such small neural networks, we propose an improved attack-sharing loss function to cope with imbalanced learning. We demonstrate that our proposed solution can detect network attacks with an F1 score above 99% for various attacks found in current intrusion detection system datasets, focusing on IoT device communication. We further show that our solution can reduce the false negative detection rate of our proposed shallow network and thus further improve network security while enabling processing at line rate in low-complexity network intrusion systems.

Keywords: network intrusion detection system (NIDS); neural network; machine learning; network security; network attack detection

check for updates

Citation: Ehmer, J.; Savaria, Y.; Granado, B.; David, J.-P.; Denoulet, J. Network Attack Classification with a Shallow Neural Network for Internet and Internet of Things (IoT) Traffic. *Electronics* **2024**, *13*, 3318. https://doi.org/10.3390/electronics13163318

Academic Editor: Christos J. Bouras

Received: 29 June 2024
Revised: 1 August 2024
Accepted: 13 August 2024
Published: 21 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The last few decades have shown significant growth in the interconnectedness of almost all parts of life. The proliferation of ever-faster wired and wireless network connections combined with advances in energy-efficient and powerful mobile processors has led to a massive deployment of ubiquitous Internet of Things (IoT) devices.

The COVID-19 pandemic in 2019, with its restrictive social distancing measures, accelerated the push towards a much more digitalized society [1]. In weeks, the whole economic sector and most of the education sector were forced into predominantly digital interactions. Mobile healthcare services were also swiftly put in place. The long-lasting impacts of these transformations are hard to predict, but digitalization will most probably continue to occur.

The transition from a pre-digitalized society towards a mostly digital one yields tremendous opportunities for improved and more economical services. However, the

dependence on digitally connected devices and services renders us susceptible to attacks on the underlying network and its connected appliances.

Attackers of the systems above can be classified into three attack scenarios [2]. Based on the attacker's motivation, resources, and the complex nature of the attack, it is possible to distinguish between cybercrime, cyber-terrorism, and cyber-warfare. Cybercrime is often motivated by financial incentives. Cybercriminals often indulge in ransomware or phishing attacks. Cyber-terrorism aims to spread terror to push some form of ideological agenda. Cyber-warfare is funded or executed by nation-states and often shows high sophistication.

The presence of cyberattacks makes it necessary to implement mechanisms to protect digital infrastructures and ensure the secure and continuous operation of the corresponding services. Hardening digital services to protect them from malicious activities can take many forms, such as intrusion detection systems (IDSs) that analyze network traffic to detect suspicious activities or intrusion prevention systems (IPSs) that can act upon the detection of a potential attack.

Traditional intrusion detection systems use specially crafted patterns to match certain network traffic characteristics to detect malicious activities. Such systems produce deterministic and dependable results, but the maintenance and creation of the patterns take lots of time and effort. Furthermore, a growing number of detection patterns increases the workload of the underlying network infrastructure and can reduce throughput. In [3], machine learning algorithms can learn complex patterns reliably while detecting and classifying various attack types.

Another advantage of machine learning algorithms is their inherent capability to generalize and thus correctly classify an input, even if it differs from the initially provided training data.

Since about 2012 [4], there has been a renewed interest in machine learning. This renewal in interest was mainly driven by the development of new training methods like stochastic gradient descent and by advances in graphic processing units (GPUs) capable of dealing with computationally complex tasks like the training of deep neural networks[5]. Advances in graphical object detection and natural language processing driven by ever deeper neural networks led to the now conventional wisdom that good results in machine learning is only possible with deep and complex neural architectures. However, these deep neural networks pose a sizeable challenge when using them in edge computing devices or with high-throughput applications, such as switches in a network infrastructure. Both applications (edge devices and switching infrastructure) came with their own set of specific challenges. Modern switching infrastructure has to cope with high throughput and short latencies. Limiting the computation spent for each packet is necessary to satisfy those requirements.

Embedded systems are often subject to tight energy budgets and possess limited processing power. An example of embedded systems that can profit from machine learning-powered attack detection algorithms is distributed autonomous systems like satellite constellations as described in [6]. Detecting attacks on such connected edge devices can reduce reaction times and thus help to improve network security. However, using deep neural networks on edge platforms is very challenging, and applications are often transferred to some form of cloud server. Communication with the necessary cloud services generally consumes a considerable amount of energy and imposes additional processing latencies and financial costs for the communication services. In addition, securing the interaction with the cloud provider is necessary, which further increases the processing and communication burden on the edge device. In a high-throughput environment, calculations must be performed quickly. Deep neural networks may perform millions of computations for a single inference step, which would take a long time to complete on low-power edge devices. By contrast, using more powerful processors is costlier and generally incompatible with edge devices' power and energy budget.

In this paper, we show that a deep neural network may not be necessary to treat the problem of network flow analysis for attack detection. The reported results show that a

shallow neural network can perform equally well as competing deeper neural architectures. Using a shallow network reduces the overall processing cost and energy consumption. An additional advantage of a shallow neural network is its low computational latency, which can be further improved by parallelization.

Our main objective in this article is to analyze the traffic of a computer network to detect malicious activities. We use flow statistics that characterize the network traffic to achieve this goal. Typical characteristics of network flow entries are the mean packet size, the mean time between two packets, the total number of packets exchanged, or the number of packets per second. The advantage of using network flow characteristics is that they do not contain packet payloads, which are often encrypted and can combine multiple elementary packets into a single set of characteristics, further limiting the amount of data to analyze.

Typical attacks that can be detected are, for instance, denial-of-service attacks, where the attacker tries to bind a maximum of server resources to prevent the victim from servicing other clients [7]. Another type of attack is a port scan. In this attack, the attacker tries to find open ports on the victim's server. When the attackers find such an open port, they check in their database whether known vulnerabilities are associated with them. When attackers successfully identify a vulnerable system, they try to exploit the vulnerability to compromise the corresponding system. Yet another type of attack consists of guessing a username and password for a standard service like FTP, SSH, or any other type of restricted access.

Our objective is to find the best fully connected neural network to detect and classify such types of attacks on edge devices.

The main contributions of the present paper are as follows:

- Proposing a shallow neural network capable of detecting and classifying network attacks for various network applications.
- Proposing an improvement of the attack-sharing loss function proposed by Dong et al. [8].
- Proposing a straightforward architecture search strategy for a low-complexity neural network.
- Showing the applicability of our approach with different intrusion detection training datasets representing general Internet use as well as Internet of Things and industrial applications.

Section 2 presents the datasets used for this article. This section contains a detailed description of the CIC-IDS2017 and its derived Lycos dataset, and a brief overview of additional datasets used for extended testing of our proposed solution. In Section 3, we explore the problem of imbalanced learning, which is associated with all significant intrusion detection training datasets. Based on the observed problem of imbalanced learning, we will provide an overview of possible mitigation strategies and present our proposed improvement on the attack-sharing loss function, proposed by Dong et al. [8]. Section 6 introduces the different metrics used to determine a neural network's performance. In Section 7, we present the performances achieved with our proposed solution. We will discuss our results in Section 8. Finally, Section 9 summarizes the paper's main contributions and concludes this work.

2. The Training Dataset

2.1. The CIC-IDS2017 Dataset

For a deep learning algorithm to be trained successfully, it is necessary to use a sufficiently large and detailed training dataset. Problems in the training data, such as biases or misclassifications, are later reflected in the trained neural network. That is why creating a suitable training set is a critical task. There is a variety of different network intrusion training datasets available for the scientific community to use. This article uses the CIC-IDS2017 dataset that the Canadian Institute for Cyber Security freely provides.

A detailed description of the dataset and a comparison with other available datasets is given in [9]. The CIC-IDS2017 dataset is based on a realistic setup consisting of a small network with several server and client computers and another network that is the source

of attacks. The researchers of the Canadian Institute for Cybersecurity (CIC) performed with this setup various network attacks over the timespan of a week. The captured network traffic was then converted into a series of network flows. The flow generation was performed using the CICFlowMeter tool, also developed at the Canadian Institute for Cybersecurity. In the last step, the researchers labeled each flow with either the benign class or one of the 14 attack classes. Table 1 shows the 15 classes (14 attacks and the benign case) present in the CIC-IDS2017 dataset. In addition to the different classes, Table 1 also shows the corresponding number of flow entries.

Table 1. CIC-IDS2017 and Lycos-IDS2017 attack classes and number of samples.

Class	CIC-IDS2017	Lycos-IDS2017
BENIGN	2,273,097	1,395,675
DoS Hulk	231,073	158,988
PortScan	158,930	160,106
DDoS	128,027	95,683
DoS GoldenEye	10,293	6765
FTP-Patator	7938	4003
SSH-Patator	5897	2959
DoS slowloris	5796	5674
DoS Slowhttptest	5499	4866
Bot	1966	735
Web Attack Brute Force	1507	1360
Web Attack XSS	652	661
Infiltration	36	-
Web Attack Sql Injection	21	12
Heartbleed	11	11

2.2. Problems of the CIC-IDS2017 Dataset

During our work with the CIC-IDS2017 dataset, we observed various issues concerning different aspects of the generated network flow data. The "Flow Byte/s" and "Flow Packet/s", feature fields contained infinite values. The "Fwd Header Length" column was duplicated. A total of 8 features had all zero values. Ten features contained negative values even though the represented flow feature did not allow negative values. Examples of features containing negative values are "Flow Duration" and "Init Win Bytes Forward". Further literature analysis revealed the availability of a corrected version of the CIC-IDS2017 dataset called Lycos, published by Rosay et al. [10]. This version of the dataset addresses the shortcomings mentioned above.

2.3. The Lycos Dataset

Rosay et al. [10,11] describe the different issues of the CIC-IDS2017 dataset in their article. In response to those errors, they developed a new flow extraction and labeling tool called LycoSTand. This new tool aims to create a corrected dataset that can serve as a standard for future machine learning for intrusion detection. The Lycos-IDS2017 dataset contains 78 features and the class label. Table 1 shows the various class labels and the corresponding number of flow entries in the Lycos-IDS2017 dataset.

Comparing the two leftmost columns of Table 1 shows that the number of flow entries of the Lycos-IDS2017 dataset is lower than that of the initial IDS-2017 dataset. According to [10], the decrease in flow entries can be attributed to corrections in the flow extraction process. Comparing the number of flows within the dataset shows that the majority of the samples in the dataset are of the benign class.

2.4. Dataset Simplification

The data in Table 1 reveals a strong imbalance of the different attack classes. To mitigate the effects of this imbalance, we followed the simplification strategy proposed by Boxiang Dong et al. [8]. In their paper, the authors propose combining attacks of a similar family and dropping those severely underrepresented. Figure 1 illustrates this simplification process schematically.

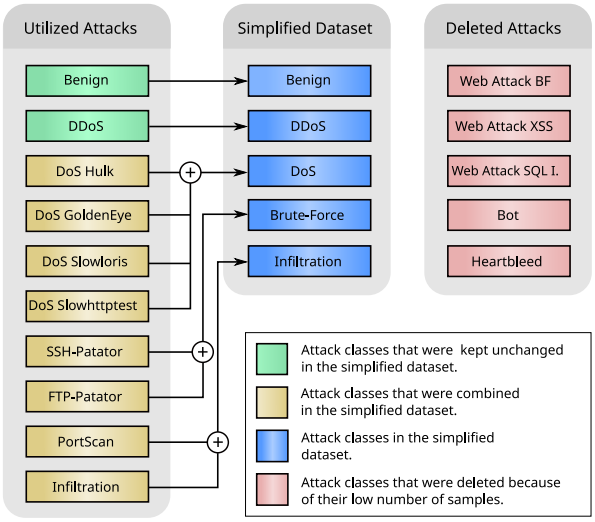

Figure 1. Schematic representation of the dataset simplification process

A detailed description of the sample distribution of the simplified dataset is shown in Table 2. The obtained simplified dataset exhibits a much more balanced distribution, but still contains a strong difference in the number of samples between the majority class (the benign traffic) and the minority class (the Brute-Force attack).

Table 2. Distribution of the simplified dataset.

Class	Samples
Benign	1,395,675
DoS	176,293
Infiltration	160,106
DDoS	95,683
Brute-Force	6962

We included the following intrusion detection training datasets to expand our proposed solution's applicability. The advantage of those datasets is their inclusion of smart devices that are becoming an increasingly important aspect of today's network landscape.

2.5. Bot IoT Dataset

The Bot IoT [12–14] intrusion detection dataset, created at the University of New South Wales (UNSW), has about 73 million entries. The authors also proposed a subset containing about 3 million flow entries to obtain a more manageable dataset. Each flow entry of the 5% reduced dataset contains 46 features. Furthermore, the provided data comprise 11 flow categories, of which one is the benign category of regular network traffic. The dataset consists of different attack scenarios on various smart devices, such as weather stations, smart fridges, automated garage doors, and motion-activated lights communicating via the MQTT protocol with a simulated cloud service.

2.6. WUSTL-IIOT-2021 IDS Dataset

The WUSTL-IIOT-2021 [15,16] simulates an industrial installation of connected sensors and actuators. The dataset comprises about 1 million flow entries with 49 flow features and 5 traffic categories. The communication between the different smart devices was realized via the Modbus protocol. Figure 2 shows the distribution of samples in each intrusion detection training dataset.

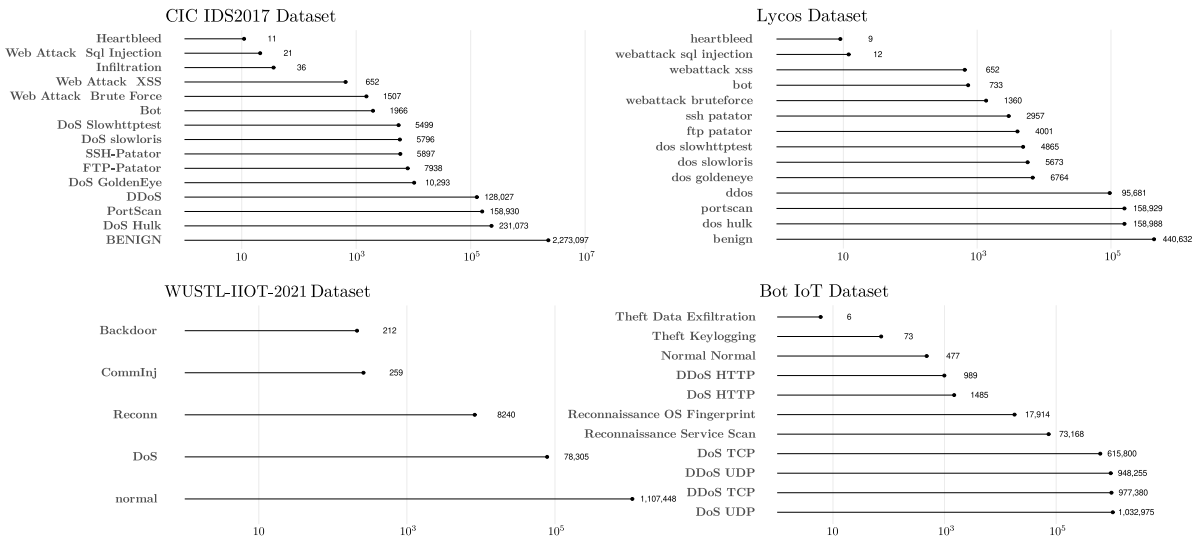

Figure 2. Overview of the class distribution of the different training datasets.

The distributions depicted in Figure 2 show that all four intrusion detection training datasets are heavily imbalanced. The imbalance lies in the fact that some traffic classes have much fewer samples than others. In the datasets CIC-IDS2017, Lycos, and WUSTL-IIOT-2021, the imbalance lies mainly between the normal traffic and attack data. The Bot IoT dataset has a strong imbalance between some attacks and the normal traffic and the much stronger represented DoS and DDoS attacks.

Such a strong imbalance in the datasets can harm the training and, thus, the performance of a neural network. In the next section, we will explore the effect of imbalanced training datasets in more detail and propose a solution to the challenges an imbalanced learning task poses.

3. Imbalanced Learning

From Table 1 and Figure 2, it is obvious that attacks are relatively rare events in the training dataset. Training neural networks on such imbalanced data poses several challenges. The problem is that a machine learning algorithm in a supervised learning environment can classify all inputs as belonging to the majority class and produce a low global error rate. In this case, one could observe good results for some basic accuracy metrics, even if the actual performance of the trained system is quite poor for some less frequently observed classes. There are numerous strategies to deal with the problem of imbalanced training data. A comprehensive overview of such strategies can be found in [17]. One class of such strategies consists of changing the dataset to shift the distribution of samples towards a more balanced one. One possible way of achieving this is to copy samples of the minority class via an oversampling algorithm [18]. It is also possible to follow the opposite strategy and remove samples from the majority class via undersampling [19]. Some strategies combine oversampling and undersampling algorithms, such as the Difficult Set Sampling Technique (DSSTE) proposed by Liu et al. [20]. Another method to deal with imbalanced data that does not change the underlying dataset is using specialized loss functions like the attack-sharing loss proposed by Dong et al. [8].

In the present article, we use an improved version of this loss function, which we will describe in the following section.

Improved Attack-Sharing Loss

To address the problem of imbalanced learning, Dong et al. [8] proposed an adaptation of the cross-entropy loss function, which they called attack-sharing loss, that can be quantified with Equation (1).

$$J_{AS} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N} \lambda\left(I(y^{(i)},1) log\ p_1^{(i)} + \sum_{j=2}^{c} I(y^{(i)},j)\ log\ (1-p_1^{(i)})\right)\right] \quad (1)$$

J_{CE} is the vanilla cross-entropy loss function, N is the number of training samples in the batches, and c represents the number of classes in the training dataset. The factor λ scales the effect of the regularizing term. A small value of λ makes the loss function behave almost like the vanilla cross-entropy function. $y^{(i)}$ is the i-th label and $p_1^{(i)}$ the i-th predicted probability for the first (majority) class. The function $I(a,b)$ is the indicator function defined by Equation (2).

$$I(a,b) = \begin{cases} 1 & if\ a = b \\ 0 & otherwise \end{cases} \quad (2)$$

To further tune the regularization term of the original attack-sharing loss function, we introduce two new parameters: α and β. Our aim with this new loss function (3) is to reduce the false negative rate even more by increasing the regularization penalty contribution in the case where the current sample belongs to the minority class.

$$J_{AS} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N}\left(\alpha\left(I(y^{(i)},1) log\ p_1^{(i)}\right) + \sum_{j=2}^{c} s_j\ I(y^{(i)},j)\ log\ (1-p_1^{(i)})\right)\right] \quad (3)$$

The factor α scales the penalty contribution for the case where the current training sample belongs to the majority class. The second parameter β is multiplied by the inverse frequency of the current sample's minority class, resulting in the scaling factor s_j as shown in (4).

$$s_j = \beta * (1 - \frac{n_j}{N_{mc}}) \quad (4)$$

In Equation (4), n_j represents the number of samples in the current batch belonging to the minority class j, N_{mc} represents the total number of minority samples in the batch. The scaling factor s_j thus reflects the distribution of the minority classes in the current batch. Rare samples in the batch cause a higher regularization penalty, which in turn causes a stronger displacement of the decision boundary towards the attack classes. The two factors α and β are hyperparameters, allowing a more nuanced training process tuning. In our tests, for which detailed results are reported later, we achieved good results with an empirical value of 5 for α and 20 for β.

4. Neural Network Architecture Search

We aim to define a neural network adapted to detect attacks on the edge. We choose to conceive a multi-layer perceptron (MLP), a well-known model currently used in detection problems. This kind of neural network conceptually consists of multiple layers of artificial neurons. The first layer, the input layer, consists of neurons whose only purpose is to pass the provided input data to the next layer. The input layer has one dedicated neuron for each input feature.

Multiple layers of neurons follow the input layer. Those are called hidden layers because they lack direct connections to the input. The last layer of an artificial neural network consists of neurons that produce the network's output. Traditionally, there are as many such neurons as classes in a classification task. The output neurons produce values between 0 and 1 that can be interpreted as a probability that the presented input describes a

sample of the corresponding class. The overall structure of such an artificial neural network is called a neural network architecture.

4.1. Overview of Architecture Search Strategies

There are various possible strategies to find good and effective neural network architectures. It is, for instance, possible to use a grid search algorithm [21] to explore the possible solution space. Possible parameters of a grid search are the width (number of neurons in each layer) and depth (number of layers) of the neural network. A corresponding network is built and trained for each value of such a grid. The best-performing network is then picked. An advantage of this strategy is that it is relatively easy to implement. However, training a potentially large number of neural networks is quite time-consuming.

Other strategies involve reinforcement learning or genetic algorithms [22]. Such search algorithms are time-consuming and difficult to implement. Aiming for a simple search strategy producing low-complexity networks led us to define our strategy.

4.2. Our Network Search Strategy

Our strategy is based on a variant of the grid search algorithm. It consists of iteratively increasing the size of the network architecture under test. The procedure starts with a minimal neural network comprising one input layer with one neuron per feature, one output layer with one neuron for each detectable category, and just one hidden layer initially consisting of only a couple of neurons, to which we gradually add neurons. Each network was trained, and the performance was compared to the previous architecture. We did this until the performance increase was no longer significant.

In a further step, we continued the search by adding hidden layers. We trained the network for each additional layer and compared its performance with the previous model. When the performance gain reached a level where it was not significant anymore, we stopped the search. Figure 3 shows a schematic representation of the algorithm.

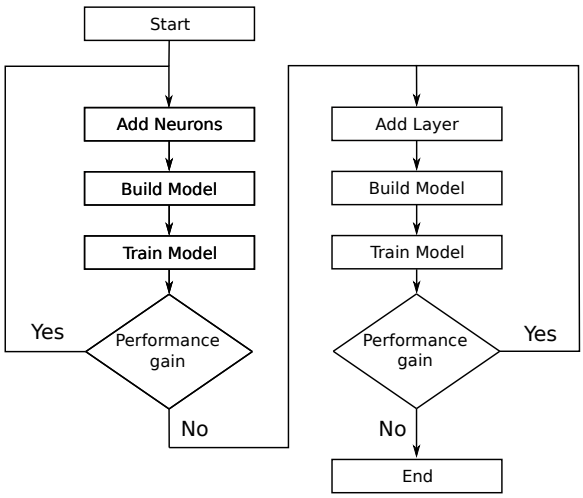

Figure 3. Schematic representation of our architecture search algorithm.

5. Implementation

We have defined a strategy to find an efficient MLP that can detect network attacks on the edge; in the next step, we implemented a data preparation pipeline to manage out-of-range dataset values such as infinite or Not a Number. We chose to replace infinite values with the corresponding column's maximum value. Missing values, indicated by the Not a Number (NaN) value, were replaced with the column's median value by applying the Scikit-learn SimpleImputer class. Further, the data were normalized by subtracting the mean value and dividing by the corresponding standard deviation. For this normalization step, we used the Scikit-learn StandardScaler class.

To improve confidence in the generalization capacity of the model, we used the Scikit-learn StratifiedKFold class for a 5-fold cross-validation. We further split the holdout set into two equal parts using the Sciekit-learn train_test_split function with the stratify option activated. The advantage of the stratified shuffle split is that the train, test, and validation datasets contain the initial proportion of samples for each class label. For the Lycos-IDS2017 dataset with all attack classes, this approach however led to the problem that for some class labels, the holdout set did not contain enough samples for the split. We therefore applied in this case a standard stratified shuffle split to obtain an 80% training, 10% validation, and 10% test dataset.

The MLP is implemented in Pytorch. We use the rectified linear unit (ReLU) as activation function for the artificial neurons in the single hidden layer. The advantage of the rectified linear unit (Equation (5)) is its very low computational complexity. The ReLU activation function can be implemented by combining a comparison and a value assignment operation.

$$f(x) = max(0, x) \tag{5}$$

We use the Softmax activation function for the output nodes, shown in Equation (6).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{6}$$

The training was performed with the standard stochastic gradient algorithm. The values of the weights and biases were initialized with the HeNormal initializer [23]. As an optimizer, we chose the Nadam optimizer from the Pytorch library, which had a learning rate of 0.001.

We used a custom early stopping to prevent overfitting that monitored the validation loss and F1 score.

6. Performance/Complexity Measures

We use the confusion matrix to judge each neural network's performance. A confusion matrix shows the results of multiple inferences typically made with the test dataset at the end of the training to evaluate the model's performance. Figure 4 shows a schematic representation of a confusion matrix for a binary classification problem. Every cell of such a matrix contains the number of samples classified according to the corresponding class. The columns of a confusion matrix represent the true value given in the test dataset. The rows represent the model's prediction. The binary case has two correct classifications: true negatives (TNs) and true positives (TPs). These values are located on the diagonal of the confusion matrix. They represent cases where the model could correctly classify the input data.

The number of negative samples wrongly classified as positive is called the number of false positives (FPs). The number of positive values wrongly classified as negative is called the number of false negatives (FNs).

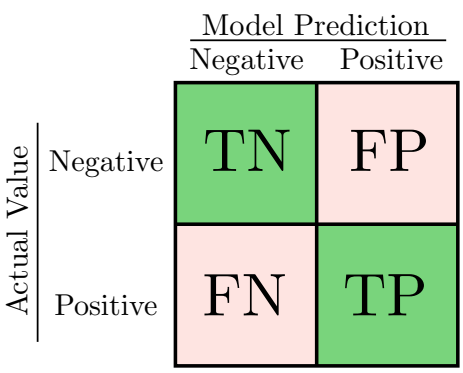

Figure 4. Schematic representation of a binary confusion matrix.

Several performance metrics exist based on the values shown in a confusion matrix. The recall metric represents a ratio of the true positive values and the total number of positive class samples, as shown in Equation (7). The recall metric can be interpreted as the capacity of the network to predict the positive class.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

The precision metric, as shown in Equation (8), is the ratio of the number of true positive predictions relative to the total number of samples classified as positive. The precision metric can be interpreted as the quality or pureness of the prediction. A precision value close to one means that only a few false positive predictions exist.

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

It is possible to derive a combined measure by applying the harmonic mean. This type of performance metric is called F1-Score, shown in Equation (9). The F1-Score always lies between the two performance metrics, precision and recall. It gives a good overall measure of the neural network's performance.

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{9}$$

For neural networks with more than two classes, it is also possible to calculate the performance using the abovementioned metrics. To obtain a global measure for precision, recall, and the F1-Score, it is necessary to calculate the individual metrics for each class first. The number of false positives for the i-th class is, in this case, the sum of all values in the column representing the corresponding class without the value at $c_{j=i,k=i}$, which represents the true positives value (TP_i). The formula for the number of false positives is shown in Equation (10), where N is the number of classes and $c_{j,k}$ the entry of the confusion matrix at row j and column k.

$$FP_i = \sum_{j \neq i, k=i}^{N} c_{j,k} \tag{10}$$

The value for the false negatives of the i-th class can be calculated by adding the values of the i-th row of the confusion matrix except for the value at $c_{j=i,k=i}$, as depicted by Equation (11).

$$FN_i = \sum_{j=i, k \neq i}^{N} c_{j,k} \tag{11}$$

From these values, it is possible to calculate the individual performance metrics. The mean value of the individual performance metrics is then calculated to obtain a global performance measure for the multi-class neural network.

7. Results

We have applied our search strategy and produced different MLPs of different sizes. We can see the results in Figure 5. The left part of the figure shows the performance measurements of precision, recall, and F1-Score for different numbers of neurons in a single layer. It shows that initially, the performance increases rapidly with the number of neurons when that number is increased. The graph shows that the network achieved very good results with only a few neurons. However, we observed that the performance still increased with additional neurons. The network achieves very good performance with 110 neurons in the hidden layer. If we increase the number of the neurons, the supplementary computational cost increases according to Equation (12), where X^h is the

number of additional neurons in the hidden layer, X^i is the number of neurons of the input layer, and X^o is the number of neurons of the output layer.

$$C_{sup} = X^h * X^i + X^h * X^o = X^h * (X^i + X^o) \tag{12}$$

This leads to necessary memory allocations of C_{sup} weights and an increase in terms of clock cycles that is proportional to C_{sup} to obtain a very small gain (less than 0.1). Therefore, we concluded that 110 artificial neurons in the hidden layer appear to be a good trade-off to minimize network complexity while maintaining a high detection rate. Following the proposition in [8] and to allow deeper architectures to be explored more rapidly, we limited the number of neurons in each hidden layer to 100. The graph shows that the performance of a network with a single hidden layer is already quite high (F1-Score of 0.9985). It is also apparent that additional layers cannot yield much better results. On the contrary, additional layers appear to degrade the neural network's performance.

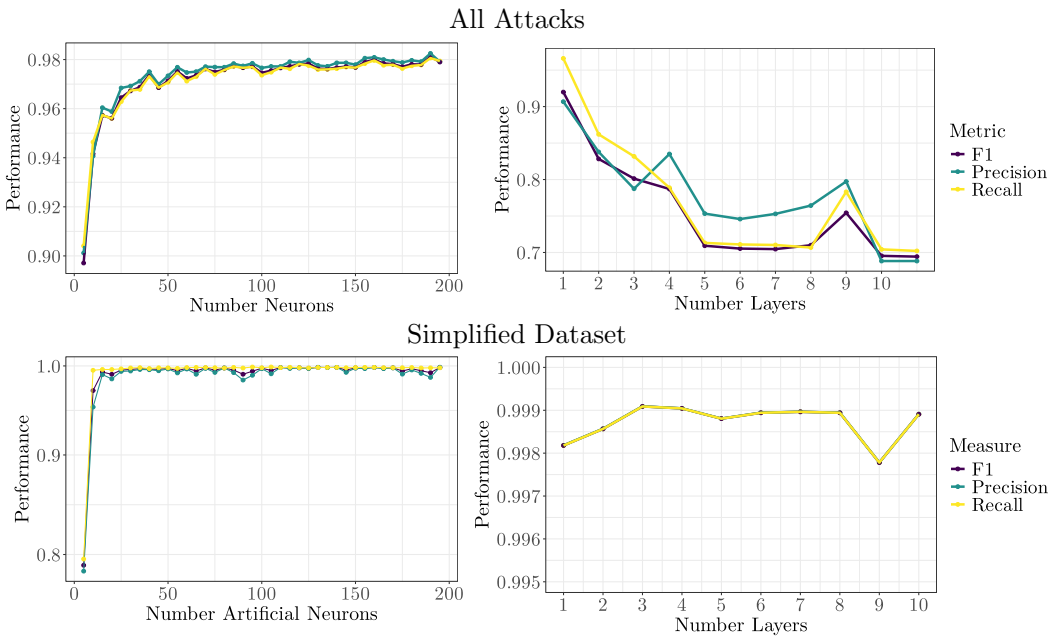

Figure 5. Results for different numbers of neurons (**left**) and hidden layers (**right**) for the complete and simplified dataset.

The graph on the right-hand side of Figure 5 shows the neural network's performance for different depths, trained on the Lycos-IDS2017 dataset with all attacks included. The single-layer network performs quite well, with an F1-Score of 0.9199. However, additional layers significantly degrade the overall performance of the network.

The two confusion matrices in Figure 6 show that our improved attack-sharing loss function improved the excellent results obtained with the original basic cross-entropy function. These results indicate that the additional regularization term successfully reduced the false negative rate of the classification caused by the disproportion of the training dataset towards the benign case. The confusion matrix on the left also shows a slight increase in false positives for the improved attack-sharing loss, representing a trade-off of our proposed solution. In terms of security, the false negative rate is more detrimental to the overall network security than the false positive rate, as falsely benign classified network traffic spreads into the network and can cause damage. These results, furthermore, show that a shallow neural network can classify the different attack classes in the simplified Lycos dataset. Based on this MLP architecture exploration, we chose a single-layer MLP. Table 3 shows the performance measurements of our single-layer MLP trained with the original simplified CIC-IDS2017 dataset and the simplified version of the improved Lycos-IDS2017 dataset (first two rows). We compare our results with those published by Boxiang Dong

et al. [8] for a network with 10 layers of 100 neurons each. The reported results are the mean results of the published individual scores, since Dong et al. did not publish a global performance measurement.

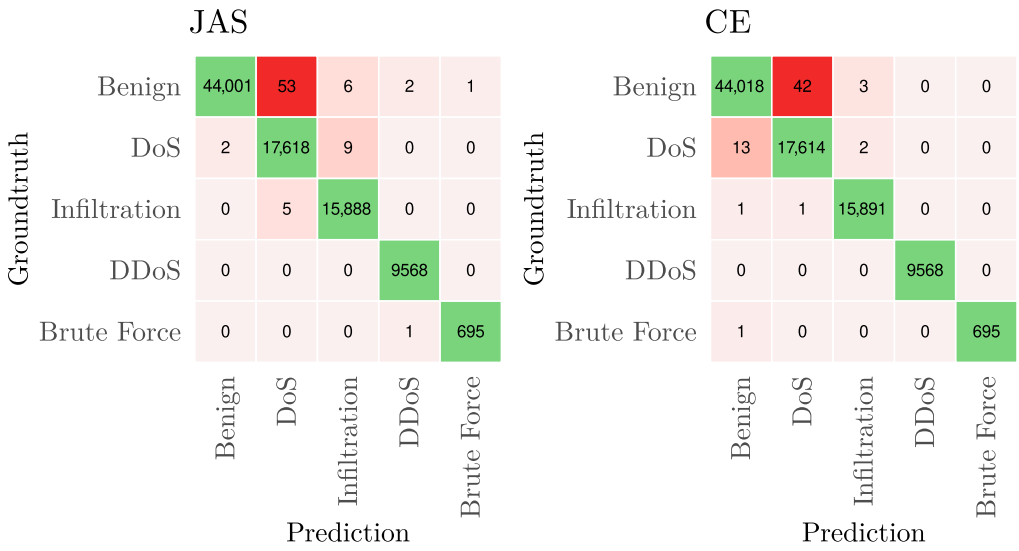

Figure 6. Comparison between the improved attack-sharing (IAS) loss function and the basic cross-entropy (CE) loss function.

The results in the first two rows show that the improved Lycos-IDS2017 dataset yields better results than the original CIC-IDS2017 dataset. A second result that is shown in Table 3 is that our shallow neural network can substantially outperform the much deeper architecture.

Table 3. Results for Lycos-IDS2017 and CIC-IDS2017 simplified dataset.

	Precision	**Recall**	**F1**
Our results on the Lycos-IDS2017 dataset	0.9981	0.9989	0.9985
Our results on the CIC-IDS2017 dataset	0.9945	0.9948	0.9946
Boxiang Dong et al. on the CIC-IDS2017 dataset [8] [1]	0.5766	0.6177	0.5965

[1] values are calculated from published individual class scores.

Table 4 shows a comparison between our results obtained with a shallow neural network consisting of a single hidden layer with 110 ReLU-activated artificial neurons and the results proposed by Rosay et al. [10] who implemented an artificial neural network consisting of two hidden layers, one of 64 and a second one of 32 SELU-activated neurons. To obtain the results in Table 4, we trained our proposed neural network on the complete Lycos-IDS2017 dataset without simplification. This way, all attack classes participated in the training.

The first row of Table 4 contains the mean values of the individual performance measurements, as described in Section 6. The second row shows the performance of the aggregated confusion matrix. To calculate these values, we combined the results of all attacks while keeping the benign case separate. This way, we obtained a binary confusion matrix that distinguishes only the two cases, benign and attack. The third row of Table 4 shows the performance measurements of the aggregated confusion matrix as they were presented by Rosay et al. [10]. Comparing Table 4 results show that our network could successfully classify network flows with an average F1-Score of about 92% If we compare our shallow MLP solution with the two hidden layers MLP from Rosay et al. [10], rows 2 and 3, we see that they achieved similar performances. They obtained around 99% accuracy, confirming that shallow MLPs can classify network attacks as effectively as

deeper networks. An advantage of a single hidden layer is that the entire hidden part of the inference step can be efficiently parallelized and accelerated. In addition, our shallow MLP uses the ReLU activation function. This activation function can be very effectively implemented. Rosay et al. use the SELU function, which implies calculating an exponential function in combination with several floating-point multiplications. Those mathematical operations are computationally very costly. Thus, our algorithm would run faster on most embedded systems.

Table 4. Results for the Lycos-IDS2017 dataset that isolate all attack types.

	Precision	**Recall**	**F1**
Our result (mean of the individual results)	0.9070	0.9661	0.9199
Our result as binary classification	0.9992	0.9984	0.9988
Rosay et al. [10]	0.9989	0.9954	0.9967

Figure 7 shows the confusion matrix for the test results of our proposed neural network for the simplified Lycos-IDS2017 dataset (left side) and the complete Lycos-IDS2017 dataset (right side). For the simplified dataset, it shows that we achieved very good results, with only a few samples wrongly classified.

The confusion matrix for the complete dataset with all attacks shows that most attack classes were very well classified. The "webattack_bruteforce" class, however, was less well detected. Difficulties to detect this attack class were also reported in [24]. Another aspect highlighted in the confusion matrix is that only a few of the provided attack samples were classified as benign. The small values visualize this in the first column of the confusion matrices shown in Figure 7. The small number of attacks wrongly classified as benign indicates a low false negative rate, which is important for an intrusion detection system, as already explained.

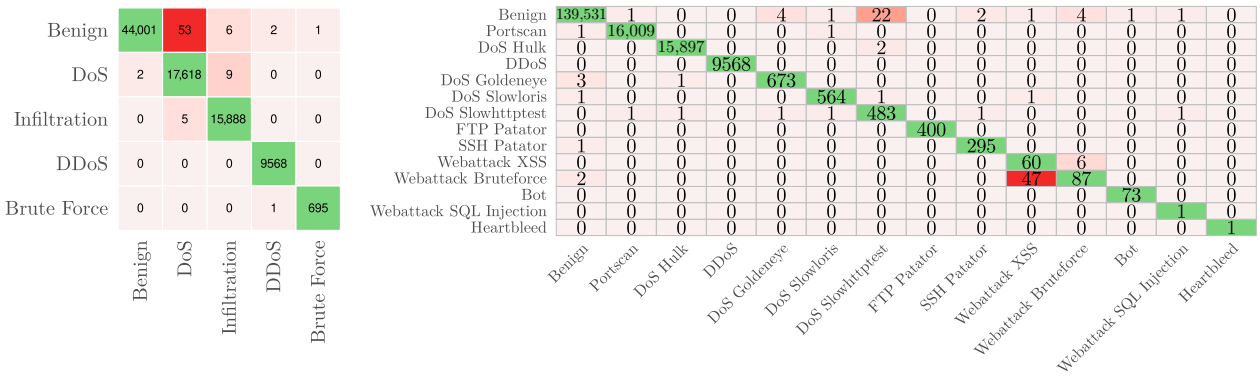

Figure 7. Confusion matrix for the simplified dataset (**left**) and the complete dataset (**right**).

Table 5 shows the individual precision and recall performance measures for each class of the complete Lycos-IDS2017 dataset obtained with our shallow neural network with 110 ReLU-activated neurons in a single hidden layer.

Table 5. Detailed performance metrics for the complete Lycos-IDS2017 dataset

Class	Precision	Recall	F1-Score
benign	0.9999	0.9997	0.9998
bot	0.9865	1.0000	0.9932
ddos	1.0000	1.0000	1.0000
dos_goldeneye	0.9926	0.9941	0.9934
dos_hulk	0.9999	0.9999	0.9999
dos_slowhttptest	0.9508	0.9918	0.9709
dos_slowloris	0.9947	0.9947	0.9947
ftp_patator	1.0000	1.0000	1.0000
heartbleed	1.0000	1.0000	1.0000
portscan	0.9999	0.9999	0.9999
ssh_patator	0.9933	0.9966	0.9950
webattack_bruteforce	0.8969	0.6397	0.7469
webattack_sql_inection	0.3333	1.0000	0.5000
webattack_xss	0.5505	0.9091	0.6858

Table 6 compares our results for two additional intrusion detection datasets, WUSTL-IIOT-2021 and Bot IoT, with performance scores reported in the literature. All three considered datasets were used to find the structure that produced the best performance with our proposed architecture and search algorithm. The best classification performance was obtained with approximately 110 neurons in all cases. Therefore, we chose the same model size of 110 artificial neurons for all datasets. We trained a model for each intrusion detection dataset, utilizing early stopping to prevent overfitting. Figure 8 shows the receiver operating characteristic (ROC) curve for the corresponding datasets. The curves show that we achieved excellent classification results for the Lycos-IDS2017 and Bot IoT datasets. However, the curve for the WUSTL-IIOT-2021 dataset shows that the model had difficulties classifying the backdoor attacks. This particular attack type had the smallest number of samples in the training dataset, which may have contributed to the decreased classification performance for this specific flow category.

Table 6. Comparison of our results with MLP implementations in the literature for IoT-centered training datasets.

	Precision	Recall	F1 Score
WUSTL-IIOT-2021 Dataset [15]			
Our Result	0.9689	0.9488	0.9564
Gaber et al. [25]	0.882	0.792	0.832
Alani et al. [26]	-	-	0.9994
Gaber et al. [25] K-NN	0.946	0.94	0.942
Gaber et al. [25] RF	0.954	0.958	0.956
Bot IoT Dataset [13]			
Our Result	0.9991	0.9818	0.9899
Koroniotis et al. [27,28]	0.999	0.999	0.999
Alsamiri et al. [29]	0.88	0.84	0.83
Alsamiri et al. [29] K-NN	0.99	0.99	0.99
Alsamiri et al. [29] RF	0.97	0.97	0.97

For the training datasets from Washington University in St. Louis and the UNSW Institute for Cybersecurity, we obtained an F1-Score well above 90 percent.

Our F1-Score of 0.9564 for the WUSTL-IIOT-2021 dataset is well above the F1-Score of 0.832 reported by Gaber et al. [25]. However, the authors only used the 17 most significant dataset features for their neural network in their paper. This, on its own, results in a considerable reduction in complexity but also comes with a decrease in model performance. Alani et al. [26]'s DeepIIoT architecture achieved an F1-score of 99%, but they only performed a binary classification into benign and attack traffic categories. However, classifying the nature of an attack can be advantageous when it comes to mitigation strategies, and thus might justify a slight decrease in the detection rate. Alsamiri et al. [29] reported an F1-Score of 0.83 for their multilayer perceptron implementation for the Bot IoT dataset. The authors only used the seven best-performing features. The comparison of those results with ours shows again the tradeoff between performance and accuracy. Koroniotis et al. [27,28] obtained an F1-Score with a binary classifier for the Bot IoT dataset of 0.999 with a much deeper neural network.

The bottom part of Table 6 presents the classification results for the respective datasets with the two commonly used algorithms K Nearest Neighbor (K-NN) and Random Forest (RF) as presented in the literature. For the WUSTL-IIOT-2021 dataset, our proposed solution outperformed the classification results presented by Gaber et al. [25]. We achieved an F1-Score close to the cited K-NN score for the Bot IoT dataset. Alsamiri et al. [29] also presented an F1-Score of 0.97 for the Adaboost and ID3 algorithms. It is also worth noting that the K-NN algorithm has the highest computational cost and, therefore, the longest runtime.

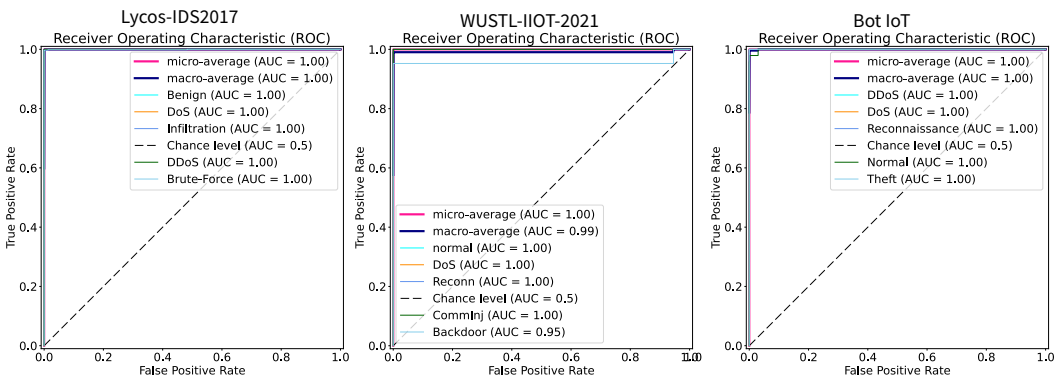

Figure 8. Receiver operating characteristic (ROC) curve for the Lycos-IDS2017, WUSTL-IIOT-2021 and Bot IoT datasets.

8. Discussion

This article presents several contributions to finding an optimal MLP to detect and classify network attacks on the edge. We have proposed an MLP search strategy consisting of an iterative approach to successively add neurons and layers to the MLP, which is then trained and its performance evaluated to find a neural architecture that, on the one hand, maximizes performance and, on the other hand, minimizes the computational cost of the network. This search strategy led to the surprising result that an MLP with a single hidden layer performs the best in the researched task. This finding contrasts the widely accepted wisdom that only deep neural networks can perform well on complex tasks.

To deal with the inherent imbalanced nature of attack detection in digital communication networks, we have improved the special attack sharing loss function [8] to further reduce the number of false negatives. The false negative rate is an essential characteristic of an intrusion detection system because attacks labeled as benign traffic cannot be further processed to refine and validate (or invalidate) the detection eventually. We have used a shallow MLP generated through a proposed simple search strategy to analyze different datasets representing different communication protocols on the Internet and in an IoT context.

We have shown that our shallow MLP performs well and could perform even better on the corrected CIC-IDS2017 dataset proposed by Rosay et al. [10], which further underlines the importance of well-designed training datasets in machine learning. It achieved a 99.85% F1-Score on the much more balanced simplified subset of the Lycos-IDS2017 dataset. This is a very good result that allows us to consider implementing our shallow MLP in an actual intrusion detection system.

On the considerably imbalanced complete Lycos-IDS2017 dataset, we achieved an F1-Score of 91.99%, which is significantly lower than the results of the simplified dataset. However, we calculated the overall performance measures as the mean value of the individual performance metrics. Such an approach makes the resulting global measure quite sensible to the performance of the minority classes. We could also have used a weighted mean (as provided in the Scikit-learn library) to calculate the global metrics, which would have resulted in a higher performance score because of its emphasis on the majority classes. Nevertheless, this would have hidden that the model struggles with some minority classes. Our presented results thus give a more conservative judgment of the actual model performance.

When the objective of the neural network is more focused on detecting an attack than the actual classification, it is advantageous to consider measurements based on the aggregated confusion matrix. Comparing the reported results of the last two rows of Table 4 shows that we were able to further improve on the performance published in the literature.

Another objective was to evaluate the capacity of our proposed neural network solution to classify network traffic in different networking scenarios. To validate our approach, we chose various network intrusion detection training datasets. The CIC-IDS2017 and its derived Lycos dataset represent a more general networking scenario of everyday Internet usage. In contrast to those two datasets, the WUSTL-IIOT-2021 and Bot IoT datasets include different types of smart devices and industrial applications. Those additional networking applications introduce new network protocols and connection patterns to the network traffic classification and attack detection task. Our results have shown that the proposed shallow neural network can detect attacks with a high F1-Score for both networking scenarios.

The presented results encourage us to extend our research in the future to further networking scenarios and to an implementation of our proposed neural network for the detection of network attacks in an embedded environment. Exploring additional model compression techniques like bit reduction or feature search and engineering also seems promising for successfully implementing our neural network on an embedded device. Another promising future research direction is combining our classification solution with an anomaly detection algorithm that could trigger alerts for possible threats not currently detected by the proposed network. An additional interesting research direction will be the combination of our approach with secure communication using Blockchain technology [30].

9. Conclusions

Network services have become economically and socially crucial for our society. The advancement of digitalization is projected to only increase in the coming decade. Alongside this increase in connected devices and services, the number of attacks on digital infrastructure is projected to grow equally. The rise of extremism and global tensions further fuels this trend. In this article, we proposed a shallow neural network to detect and classify attacks on digital communication infrastructure. Successful detection of an ongoing attack is the goal of every security mechanism.

We have shown that a straightforward architecture search can yield good results in obtaining a neural network architecture that is sufficiently small to limit the computational requirements of high-throughput or embedded devices. We have further shown that a shallow neural network with 110 ReLU-activated artificial neurons can detect and classify network attacks with an F1-Score well above 90%.

To counteract the impact of imbalanced learning caused by a strong imbalance of the sample distribution in the training dataset, we proposed an improvement on the attack-sharing loss function proposed by Dong et al. [8]. We have shown that the proposed specialized loss function can decrease the number of false negatives and thus improve the overall detection rate of the model.

We have further shown that our proposed MLP can successfully be applied to different networking scenarios realized in the different training datasets used in this paper. Our results have shown that a shallow neural network is equally capable of detecting attacks in normal Internet usage and smart device communication networks. This finding is important because it shows that our proposed solution is well-adapted to various communication protocols and access patterns.

Author Contributions: Conceptualization, B.G. and Y.S.; methodology, J.E.; software, J.E.; validation, J.E.; formal analysis, J.E.; investigation, J.E.; resources, Y.S., J.-P.D. and B.G; data curation, J.E.; writing—original draft preparation, J.E.; writing—review and editing, Y.S., B.G., J.-P.D. and J.D.; visualization, J.E.; supervision, Y.S., B.G. and J.-P.D.; project administration, Y.S.; funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the NSERC Kaloom-Intel-Noviflow Industrial Chair of Professor Savaria IRCPJ-548237-18 CRSNG, by Polytechnique Montreal, and by discovery grant RGPIN-2019-05951 CRSNG (AV: 05295-2014/6574-09) to one of the authors.

Data Availability Statement: The datasets used in the paper are publicly available to everyone and can be accessed at https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 4 March 2024), https://lycos-ids.univ-lemans.fr/ (accessed on 4 March 2024), https://www.cse.wustl.edu/~jain/iiot2/index.html (accessed on 4 March 2024), https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 4 March 2024).

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

Abbreviations

The following abbreviations are used in this manuscript:

MLP	Multi-layer perceptron
K-NN	K Nearest Neighbor
RF	Random Forest
ID3	Iterative Dichotomiser 3
DoS	Denial of Service attack
DDoS	Distributed Denial of Service attack
IoT	Internet of Things
IIoT	Industrial Internet of Things
IDS	Intrusion Detection System
IPS	Intrusion Prevention System

References

1. Kallegias, A.; Costabile, I.; Robins, J.C. The Corona Decade: The Transition to the Age of Hyper-Connectivity and the Fourth Industrial Revolution. In *Architecture and Design for Industry 4.0: Theory and Practice*; Barberio, M., Colella, M., Figliola, A., Battisti, A., Eds.; Springer International Publishing: Cham, Switzerland, 2024; pp. 169–183. [CrossRef]
2. Malik, A.W.; Abid, A.; Farooq, S.; Abid, I.; Nawaz, N.A.; Ishaq, K. Cyber threats: Taxonomy, impact, policies, and way forward. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 2425–2458.
3. Issa, M.M.; Aljanabi, M.; Muhialdeen, H.M. Systematic literature review on intrusion detection systems: Research trends, algorithms, methods, datasets, and limitations. *J. Intell. Syst.* **2024**, *33*, 20230248. [CrossRef]
4. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Pereira, F., Burges, C., Bottou, L., Weinberger, K., Eds.; Curran Associates, Inc.: New York, NY, USA, 2012; Volume 25.
5. Gu, J.; Zhu, M.; Zhou, Z.; Zhang, F.; Lin, Z.; Zhang, Q.; Breternitz, M. Implementation and evaluation of deep neural networks (DNN) on mainstream heterogeneous systems. In the Proceedings of the 5th Asia-Pacific Workshop on Systems, APSys'14, Beijing, China, 25–26 June 2014. [CrossRef]

6. Zhang, Y.; Wang, Y.; Hu, Y.; Lin, Z.; Zhai, Y.; Wang, L.; Zhao, Q.; Wen, K.; Kang, L. Security Performance Analysis of LEO Satellite Constellation Networks under DDoS Attack. *Sensors* **2022**, *22*, 7286. [CrossRef] [PubMed]
7. Gelgi, M.; Guan, Y.; Arunachala, S.; Samba Siva Rao, M.; Dragoni, N. Systematic Literature Review of IoT Botnet DDOS Attacks and Evaluation of Detection Techniques. *Sensors* **2024**, *24*, 3571. [CrossRef] [PubMed]
8. Dong, B.; Wang, H.; Varde, A.S.; Li, D.; Samanthula, B.K.; Sun, W.; Zhao, L. Cyber Intrusion Detection by Using Deep Neural Networks with Attack-sharing Loss. *arXiv* **2021**. [CrossRef]
9. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSp 2018, Funchal, Madeira, Portugal, 22–24 January 2018; pp. 108–116.
10. Rosay, A.; Carlier, F.; Cheval, E.; Leroux, P. From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Melbourne, VIC, Australia, 14–17 December 2021; pp. 570–575. [CrossRef]
11. Rosay, A.; Cheval, E.; Carlier, F.; Leroux, P. Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017:. In Proceedings of the 8th International Conference on Information Systems Security and Privacy, Online, 9–11 February 2022; pp. 25–36. [CrossRef]
12. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
13. Peterson, J.M.; Leevy, J.L.; Khoshgoftaar, T.M. A Review and Analysis of the Bot-IoT Dataset. In Proceedings of the 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE), Oxford, UK, 23–26 August 2021; pp. 20–27. [CrossRef]
14. Koroniotis, N. Designing an Effective Network Forensic Framework for the Investigation of Botnets in the Internet of Things. Ph.D. Thesis, UNSW Sydney, Sydney, NSW, Australia, 2020. [CrossRef]
15. Zolanvari, M.; Teixeira, M.A.; Gupta, L.; Khan, K.M.; Jain, R. Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 6822–6834. [CrossRef]
16. Zolanvari, M. WUSTL-IIOT-2021 Dataset. 2021. Available online: https://ieee-dataport.org/documents/wustl-iiot-2021 (accessed on 11 March 2024).
17. Haixiang, G.; Yijing, L.; Shang, J.; Mingyun, G.; Yuanyue, H.; Bing, G. Learning from class-imbalanced data: Review of methods and applications. *Expert Syst. Appl.* **2017**, *73*, 220–239. [CrossRef]
18. Zheng, Z.; Cai, Y.; Li, Y. Oversampling method for imbalanced classification. *Comput. Inform.* **2015**, *34*, 1017–1037.
19. Tahir, M.A.; Kittler, J.; Yan, F. Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognit.* **2012**, *45*, 3738–3750. [CrossRef]
20. Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning. *IEEE Access* **2021**, *9*, 7550–7563. [CrossRef]
21. Liashchynskyi, P.; Liashchynskyi, P. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *arXiv* **2019**. [CrossRef]
22. Wistuba, M.; Rawat, A.; Pedapati, T. A Survey on Neural Architecture Search. *arXiv* **2019**. [CrossRef]
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
24. Rosay, A.; Riou, K.; Carlier, F.; Pascal, L. Multi-Layer Perceptron for Network Intrusion Detection: From a study on two recent data sets to deployment on automotive processor. *Ann. Telecommun.* **2021**, *77*, 371–394. [CrossRef]
25. Gaber, T.; Awotunde, J.B.; Folorunso, S.O.; Ajagbe, S.A.; Eldesouky, E. Industrial Internet of Things Intrusion Detection Method Using Machine Learning and Optimization Techniques. *Wirel. Commun. Mob. Comput.* **2023**, *2023*, 3939895. [CrossRef]
26. Alani, M.M.; Damiani, E.; Ghosh, U. DeepIIoT: An Explainable Deep Learning Based Intrusion Detection System for Industrial IOT. In Proceedings of the 2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW), Bologna, Italy, 10–13 July 2022; pp. 169–174. [CrossRef]
27. Koroniotis, N.; Moustafa, N.; Sitnikova, E. A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework. *Future Gener. Comput. Syst.* **2020**, *110*, 91–106. [CrossRef]
28. Koroniotis, N.; Moustafa, N. Enhancing network forensics with particle swarm and deep learning: The particle deep framework. *arXiv* **2020**. [CrossRef]
29. Alsamiri, J.; Alsubhi, K. Internet of things cyber attacks detection using machine learning. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 627–634. [CrossRef]
30. Bhuva, D.; Kumar, S. Securing Space Cognitive Communication with Blockchain. In Proceedings of the 2023 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW), Cleveland, OH, USA, 20–22 June 2023; pp. 1–6. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.