



Article

Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT Datasets Using DL in Cybersecurity for IoT

Imad Tareq ^{*}, Bassant M. Elbagoury, Salsabil El-Regaily  and El-Sayed M. El-Horbaty

Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt

^{*} Correspondence: emadtariq1982@gmail.com

Abstract: The IoT's quick development has brought up several security problems and issues that cannot be solved using traditional intelligent systems. Deep learning (DL) in the field of artificial intelligence (AI) has proven to be efficient, with many advantages that can be used to address IoT cybersecurity concerns. This study trained two models of intelligent networks—namely, DenseNet and Inception Time—to detect cyber-attacks based on a multi-class classification method. We began our investigation by measuring the performance of these two networks using three datasets: the ToN-IoT dataset, which consists of heterogeneous data; the Edge-IIoT dataset; and the UNSW2015 dataset. Then, the results were compared by identifying several cyber-attacks. Extensive experiments were conducted on standard ToN-IoT datasets using the DenseNet multiclassification model. The best result we obtained was an accuracy of 99.9% for Windows 10 with DenseNet, but by using the Inception Time approach we obtained the highest result for Windows 10 with the network, with 100% accuracy. As for using the Edge-IIoT dataset with the Inception Time approach, the best result was an accuracy of 94.94%. The attacks were also assessed in the UNSW-NB15 database using the Inception Time approach, which had an accuracy rate of 98.4%. Using window sequences for the sliding window approach and a six-window size to start training the Inception Time model yielded a slight improvement, with an accuracy rate of 98.6% in the multiclassification.



Citation: Tareq, I.; Elbagoury, B.M.; El-Regaily, S.; El-Horbaty, E.-S.M. Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT Datasets Using DL in Cybersecurity for IoT. *Appl. Sci.* **2022**, *12*, 9572. <https://doi.org/10.3390/app12199572>

Academic Editor: Jongsub Moon

Received: 23 August 2022

Accepted: 19 September 2022

Published: 23 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 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/>).

Keywords: DenseNet; inception time; cyber security; malware detection; ToN-IoT dataset; UNSW2015 dataset; Edge-IIoT dataset; AI

1. Introduction

Today, AI has become a critical technology in information security. AI is used to spot cyber threats and potentially malicious activities, quickly analyze millions of events, and identify many hazards. By using deep algorithms, AI systems are trained to recognize patterns, detect malware, and even detect behaviours of malware or ransomware attacks. Malware or malicious software develops more quickly; it is designed to disable and gain unauthorized access to stolen sensitive information and computers. The primary motive for creating malware is to obtain financial gain.

Additionally, AV-Test Threat Report logs more than 350,000 new malware programs daily, and in 2021 the malware circulation had increased to 114,530 million. Amazingly, January 2021 alone reported 607 million malware programs [1]. Expert hackers invent sophisticated malware evasion strategies such as polymorphism [2], code obfuscations, metamorphism [3], and others, which outperform many existing anti-malware solutions. The most common malware includes backdoors, passwords, scanning, miners, DDOS, ransomware, etc. [4]. Malware detection techniques are generally categorized into static [5] and dynamic systems [6]. A static system analyzes the disassembled code to extract the opcode sequence, strings, function call graph, API sequences, etc.

A dynamic system, also known as behavioural analysis, is implemented in a controlled environment (known as the Sandbox) to trace networks or extract system calls. It is effective in identifying subliminal behaviour. This system is powerful but resource-intensive and

lazy; moreover, the malware shows restricted behaviour in determining analysis environments. Furthermore, these systems are ineffective in detecting unknown malware or new variants. [7].

Deep learning (DL) approaches are increasingly used in applications such as fraud detection, malware detection, fragmentation, image analysis, etc., enabling inputs to develop more robust cognitive representations by learning at several levels. Convolutional neural networks (CNNs) have been primarily responsible for advancements in DL. DL methods can quickly train a model with many convolutional layers and millions of parameters by learning complex features. In this study, the knowledge of CNNs was thoroughly trained on a large dataset. CNN techniques have gained importance in time-series classification (TSC) and malware detection [8,9].

This study contributes the following:

- The model is DenseNet121 modified from working on two-dimensional to one-dimensional image shapes.
- To the best of our knowledge, there has been no comprehensive study on the ToN-IoT database. In this study, we conducted a complete investigation of the ToN-IoT database (Win7, Win10, Network, and IoT) to train the DenseNet and Inception Time models; the best result was when combining the network with Windows 10, with 100% accuracy.
- We used the Inception Time model in two shapes—first with a 1D input vector shape, and second with a time-series 2D input shape—using the sliding window technique with a window size of six in the start time model on the UNSW-NB15 database. There was a slight improvement in the results with multiple categories, with an accuracy of 98.6%.
- This paper is the first to use the Inception Time model on the new Edge-IIoT dataset, and there was an improvement in accuracy to 94.94% with multiple classes.

The rest of this paper is organized as follows: Section 2 provides a summary of the background and related work. Section 3 presents a description of the dataset. Section 4 describes the background CNN architectures, outlining how the data should be processed and modelled to extract appropriate data for comparisons and discussion. The design of experiments is presented in Section 5. Finally, Section 6 analyzes the experimental results, before concluding the paper in Section 7.

2. Background and Related Work

Artificial intelligence (AI) is a discipline of computer science that simulates the human brain to identify the best feasible solution to achieve a given goal. ML is a branch of AI that uses the results of previous experiences as future instructions without being explicitly programmed. Supervised, unsupervised, and semi-supervised learning are the three main subcategories of machine learning. DL is a sub-branch of machine learning. In the last decade, much work has been done to combine these strategies to improve cybersecurity. In most application contexts, the effects of DL models are superior to those of classic machine learning (or shallow-model) methods [10,11]. The following aspects are the most noticeable differences between deep and shallow models: running time, number of parameters, feature representation, and learning capacity.

DL models consist of a variety of deep networks. Autoencoders, GANs, and RBMs have supervised learning models. Unsupervised learning models include DBNs, DNNs, CNNs, and RNNs. DL models directly learn feature representations from the source data, such as images and text, without requiring explicit feature engineering. DL approaches provide a major advantage over shallow models for huge datasets. Network architecture, hyperparameter selection, and optimization approach are the major focuses of deep learning research. The ToN-IoT, Edge-IIoT, and UNSW2015 datasets are three current datasets in cybersecurity and the Internet of things that are discussed in this paper. Cybersecurity goals include data protection, resource protection, data privacy, and data integrity. Online, there

are several risks and attacks. Fraud detection, virus detection, intrusion detection, spam classification, phishing, and firewall disabling are all common cybersecurity issues [12,13].

Several solutions have been introduced to tackle cyber-attacks in recent years, but traditional methods fail to solve such attacks based on network traffic. However, the predictive power of deep learning models—especially CNNs—is much better than that of traditional ML models. Recently, researchers have used the DenseNet network and Inception Time. Because of their outstanding performance, they have attracted much interest.

The authors of [14] presented a new privacy-preservation-based intrusion detection framework in network traffic Internet of things fog using the SAE technique. The effectiveness of an intrusion detection system based on the ANN technique was then assessed for identifying attacks and typical vectors of the ToN-IoT dataset. P. Kumar et al., in [15], suggested a trustworthy privacy-preserving secured framework for intelligent cities. The privacy module transforms data into a new reduced shape for thwarting inference and poisoning assaults using a Blockchain-based enhanced proof-of-work (ePoW) technique and principal component analysis (PCA). The intrusion detection module implements an enhanced gradient tree boosting method (XGBoost). The authors of [16] used intrusion-detection systems using deep learning models based on ANN, DNN, and RNN by combining the entire UNSW-NB1 dataset into a single CSV file so that DL models could be tested once rather than separately for each file. The dataset was labelled with attack families as new labels.

The authors of [17] used IGRF-RFE and applied the hybrid feature selection method to the UNSW-NB15 dataset IDS for MLP-based intrusion-detection systems. IGRF ensemble feature selection and recursive feature removal using MLP are the two feature-reduction phases in IGRF-RFE. Then, as a wrapper feature selection approach, recursive feature elimination (RFE) was used to progressively delete duplicate features on the decreased feature subsets. The authors of [18] suggested IDS—a new distributed ensemble design based on fog computing, combining XGBoost, k-nearest neighbours, and Gaussian naive Bayes as individual learners. Random forests then use the obtained prediction results for the final classification. The authors of [19] suggested a residual densely connected network (Densely-ResNet) to identify attacks. It was put together with the help of surviving core modules, each comprising several Conv-GRU subnets connected by wide links. Where three virtual layers (fog, edge, and cloud) are connected, the security system can detect external threats using the (ToN-IoT) database for test evaluation. The authors of [20] proposed a detailed analysis of feature sets for network assaults with relevance and predictive power. Three feature-selection procedures were used to find and rank data characteristics: chi-squared, correlation, and information gain. The characteristics were input into two machine learning classifiers to measure the accuracy of their assault detection: random forests, and deep feedforward. N. Moustafa, in [21], proposed a new test architecture for the ToN-IoT dataset, which can be used to assess AI-based security solutions. The NSX vCloud NFV platform was utilized to assist software-defined network installation, network function virtualization, and service orchestration to create dynamic test networks that allow interaction between fog, edge, and cloud layers. Four intrusion-detection techniques based on machine learning were used to validate the dataset: gradient boosting machine, deep neural networks, naive Bayes, and random forests.

The authors of [22] proposed the Edge-IIoT dataset as an experimental resource, which uses federated deep learning and centralized intrusion detection with standard assessment criteria. Three studies were conducted using binary, 6-class, and 15-class categorization to study the traffic predictability and detection effectiveness of various cyber-attacks and threat models. Four classifiers were used to achieve this: RF, SVM, kNN, and DNN. Table 1 summarizes previous studies.

Table 1. Summary of previous studies.

Auth	Dataset	Algorithm	Accuracy	Description
P.Kumar [14]	TON-IoT	ANN	99.44%	The SDIoT-Fog data are shielded using the SAE approach from inference assaults that can be produced using system-based ML techniques.
P. Kumar et al. [15]	TON-IoT	TP2SF	98.84%	The data are transformed into a new reduced shape for thwarting inference and poisoning assaults using a Blockchain-based enhanced proof-of-work (ePoW) technique and principal component analysis (PCA).
Aleesa. [16]	UNSW-NB1	DL	99.59%	The authors combined the entire UNSW-NB1 dataset into a single CSV file so that DL models could be tested once rather than separately for each file.
Yin, Y. [17]	UNSW-NB15	-	84.24%	The authors used the IGRF-RFE hybrid feature selection method to identify MLP-based intrusion detection algorithms.
P. Kumar [18]	UNSW-NB15	RF	93.21%,	To protect the IoT, XGBoost, k-nearest neighbours, and Gaussian naive Bayes are combined as individual learners. Random forests then use the obtained prediction results.
Wu, P [19]	TON-IoT	Densely-Resnet	Win7 92.99% Network 99.93%	Connected three virtual layers (fog, edge, and cloud layers) as a whole to discover external attacks more comprehensively.
M.Sarhan [20]	TON-IoT	DFE, RF	96.10% 97.35%	The authors proposed a detailed analysis of feature sets that were ideal in terms of relevance and predictive power using chi-squared, correlation, and information gain.
Moustafa, N. [21]	TON-IoT Linux	AI	—	The authors used the validation method of AI-based cybersecurity tools for malware detection, intrusion detection, and privacy protection.
M.A. Ferrag [22]	Edge-IIoT	DNN	94.67%	A new cybersecurity dataset for IoT and IIoT applications—called Edge-IIoT—was proposed for use in ML-based intrusion-detection systems.

Research into architectures that improve the accuracy and performance of CNNs has been an active area of study for some time. This research has resulted in different types of CNN models, including LeNet, AlexNET, VGGNet [23], GoogleNet/Inception, ResNet, ZFNet, MobileNets, and DenseNet [23,24]. In this paper, we compare two CNN models—the Inception Time and DenseNet models—and evaluate them using three databases to detect cyber-attacks.

3. Dataset Descriptions

3.1. ToN-IoT Dataset

The ToN-IoT dataset [25] is intended to collect and analyze mixed data sources from the IoT and IIoT, and it contains heterogeneous data collected from different sources, including telemetry data from connected devices, Windows and Linux system logs, and system network traffic. The Internet of things is compiled from a realistic network. To evaluate the accuracy and efficiency of various cybersecurity applications based on artificial intelligence, the ToN-IoT dataset is designed to connect many virtual machines, cloud layers, blur, edges, and physical systems. It dynamically bulletins these interactions using NVE, SDN technology, and service coordination. It also contains simultaneous sets of legitimate and offensive events in network systems, operating systems, and IoT services.

Furthermore, the ToN-IoT dataset is represented in CSV format with a categorized column representing the attack or normal behaviour and the type of attack subclass, which refers to nine different kinds of attacks (XSS, DDoS, DoS, password cracking attacks, reconnaissance or verification, MITM, ransomware, backdoors, and injection attacks) [12,23]. Because the data are imbalanced, we use class weights, as they give all classes approximately equal priority in gradient changes no matter how many samples we have from each class in the training data. The numbers and types of records in the ToN-IoT dataset are illustrated in Table 2.

Table 2. The types and numbers of records in the entire ToN-IoT dataset and its testing and training sets.

Type of Event	Total Data Record	Train–Test Record
Backdoor	508,116	20,000
DoS	3,375,328	20,000
DDoS	6,165,008	20,000
Injection	452,659	20,000
MITM	1052	1043
Scanning	7,140,161	20,000
Ransomware	72,805	20,000
Password	1,718,568	20,000
XSS	2,108,944	20,000
Normal	796,380	300,000
Total	22,339,021	461,043

ToN-IoT Dataset Statistics

Windows 7, Windows 10, Network, Win10–Network, and IoT datasets contain many normal and attack types. The Windows 7 dataset has 28,366 records and 132 features, whereas the Windows 10 dataset contains 35,975 records and 124 features, Network has 21,978,632 records and 42 features, and Win10–Network has 1,073,754 records of normal and attack observations. These datasets cover training and testing machine learning models, as shown in Table 3.

Table 3. The numbers of normal records and the types of attachments collected in the Win7, Win10, Network, and Win10–Network datasets.

Type of Event	Win7	Win10	Network	Win10–Network
Backdoor	1779	-	508,116	-
DoS	-	525	3,375,328	109,957
DDoS	2134	4608	508,116	498,920
Injection	998	612	452,659	24,311
Mitm	-	15	1052	87
Scanning	226	447	7,140,161	208,572
Ransomware	82	-	72,805	-
Password	757	3628	1,718,568	101,398
XSS	4	1269	21,089,844	106,746
Normal	22,387	24,871	796,380	23,763

3.2. Edge-IIoT

The cybersecurity dataset for Internet of things (IoT) and industrial Internet of things (IIoT) applications is used in intrusion-detection systems based on machine learning. IoT data are collected from more than 10 different types of devices, such as low-cost digital sensors for sensing temperature and humidity, pH sensor meters, ultrasonic sensors, heartrate sensors, water-level detection sensors, soil moisture sensors, flame sensors, etc. In this database, 14 different types of attacks involving IoT and IIoT protocols are analyzed and classified into five threats, including DoS and DDoS attacks, information gathering, injection attacks, an-in-the-middle attacks, and malware attacks. Out of 1176 characteristics, 61 are highly correlated. The 20,952,648 usual attack statistics in Edge-IIoT [26] include 11,223,940 normal and 9,728,708 attacks. [27]. We split this dataset into 20% for tests and 80% for training, with a stratification option to keep the percentages static for all classes. A total of 1,909,671 samples were taken from the dataset: 1,527,736 for the training set and 381,935 for the test set; they were distributed into 15 categories, as shown in Table 4.

Table 4. The total numbers and the different types of records in the Edge-IIoT dataset.

IoT Traffic	Type of Event	Data Record
Normal	Normal	1,091,198
	DDoS-UDP	97,253
	DDoS-ICMP	54,351
	SQL-injection	40,661
	DDoS-TCP	40,050
	Vulnerability scanner	40,021
	Password	39,946
Attack	DDoS-HTTP	38,835
	Uploading	29,446
	Backdoor	19,221
	Port-scanning	15,982
	XSS	12,058
	Ransomware	7751
	Fingerprinting	682
	MITM	286

3.3. UNSW-NB15

The Cyber Range Lab of the Australian Center for Cyber Security released this dataset in 2015, and it is frequently utilized in the research community (ACCS). For the UNSW-NB15 dataset [25], the authors used raw network packets generated by the IXIA perfect storm program. At the bottom of the test, nine attack scenarios are implemented: DoS, fuzzes, analysis, backdoor, generic, reconnaissance, shellcode, exploits, and worms. Forty-nine network traffic features were extracted, and the Argus and Bro-IDS programs were employed. There are 2,540,044 streams in the dataset, including 2,218,761 benign and 321,283 aggressive streams [17,28]. We split it into 20% for tests and 80% for training, with a stratification option to keep these percentages static for all classes. A total of 2,540,047 were taken from this dataset: 2,032,037 for the training set and 508,010 for the test set. They were distributed into 10 classes, as shown in Table 5.

Table 5. The total numbers of records in the UNSW-NB15 dataset.

Type of Event	Data Record
Normal	2,218,764
Generic	215,481
Exploits	44,525
Fuzzers	24,246
Reconnaissance	13,987
DoS	16,353
Analysis	2677
Backdoor	2329
Shellcode	1511
Worms	174

4. Structural Models

The main objective of this study was to run two deep learning models for intrusion detection and attack recognition. These models are briefly described below. DenseNets are the next stage in deepening convolutional networks' depth. DenseNet uses direct connections between any layer and the following layer to enhance the flow of information between layers. DenseNet requires fewer parameters than traditional CNNs and can reuse the features. Inception Time is the efficient use of computing resources with a minimal increase in computational overheads for the high-performance output of the starting network and the ability to extract features from input data at different scales using different-sized convolutional filters. The output of these models is evaluated by calculating accuracy, precision, recall, and loss.

4.1. DenseNet Model

This section explains the architecture of the used DenseNet. We used class weights, which helped us to optimize scores for a few categories. The dataset was split into test and training sets to avoid overfitting. As an optimization algorithm, the RMSProp optimizer was used. RMSProp's key function is to keep each moving average of the weight of squared gradients. The model is a state-of-the-art DenseNet121 model modified from working on 2D image shapes to work on 1D input vector shapes, e.g., CNN 2D to CNN 1D, or max-pooling 2D to max-pooling 1D, representing the attacks in the datasets, as described in Section 3. We used Densnet121 with depth-of-model (6, 12, 24, and 16) layers and 32 filters. The model consists of one-dimensional generic convolution layers and seven kernels with a stride of two, followed by three max-pooling kernels with a stride of two.

Each layer delivers its feature maps to the layers below and receives new inputs from the layers above it. The dense block is made up of concatenated layers. Within each block, the dimensions of the feature maps remain constant, but the number of filters varies. A dense block is made up of several interconnected layers. A single layer appears in 1D convolution, batch normalization, and ReLU activation. The transition layer joins two neighbouring dense blocks, since the feature map sizes are the same within the dense block, and the transition layer reduces the feature map's dimensions, followed by 1D global average pooling; its purpose is to gather all of the blocks or outputs from the previous block. Finally, there are fully connected softmax and output layers [29,30]. The DenseNet architecture is illustrated in Figure 1.

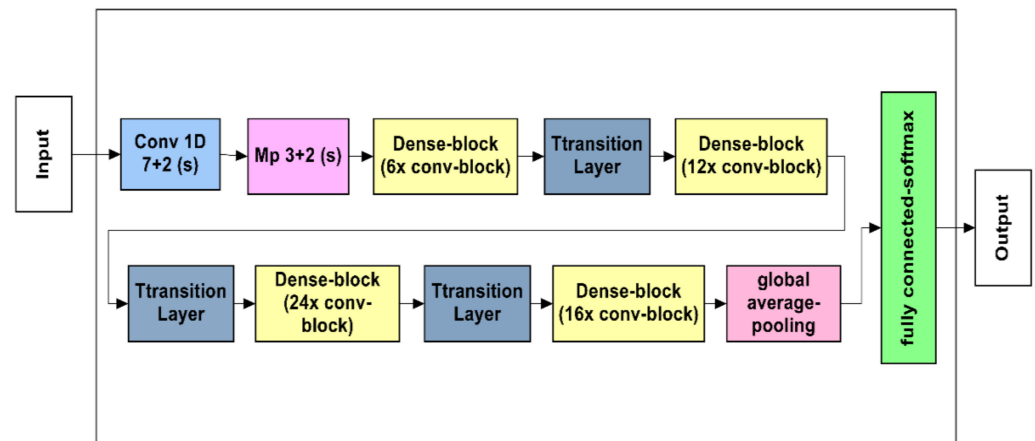


Figure 1. DenseNet121 architecture.

Each conv block consists of two convolution layers:

1. Conv_1 consists of kernel 1, stride 1, padding same, number of filters #F;
2. Conv_2 consists of kernel 3, stride 1, padding same, number of filters #4*F.

DenseNet121 Model Description

#Function Dense_Block

- Input: x = sample
 - Output: stacked layers
1. For _ in range (block_size):
 - a. layer = Conv_Block (x, 4*filters)
 - b. layer = Conv_Block (layer, filters, 3)
 - c. x = concatenate ([layer, x])

#Function Conv_Block

1. x = BatchNormalization (x)
2. x = ReLU (x)
3. x = Conv1D (input = x, filters, kernel, strides, padding = 'same')

#Function Transition_Layer

1. x = Conv_Block (x, ((x) last dimensional shape) // 2)
2. x = AvgPool1D (input = x, 2, strides = 2, padding = 'same')

#Function Desnet121

1. x = Conv1D (input, filters = 64, kernel = 7, strides = 2, padding = 'same')
2. x = MaxPool1D (input = x, kernel = 3, strides = 2, padding = 'same')
3. For block_size in block_shapes [6,12,16,24]:
 - a. s_block = Dense_Block (x, block_size, filters)
 - b. x = Transition_Layer (s_block)
4. x = GlobalAveragePooling1D (s_block)
5. Output = Dense (input = x, number_class, activation = 'softmax')

4.2. Inception Time Model

This section explains the Inception Time model architecture. After data are collected and processed, we use class weights. The dataset is split into test and training sets to avoid overfitting. The Adam optimizer is used as an optimization algorithm instead of the standard stochastic gradient descent method to iteratively update the network weights based on training data. Inception Time is used with a depth of 10 layers and 32 filters, with a short layer over 3. For each inception module, three kernels are created from $40 / (2^i)$ as i increases from 0 to 3.

In the inception network, the Inception Time model was employed in two shapes: one with a 1D input vector shape, and the other with a time-series 2D input shape utilizing the sliding window technique. Each block comprises three inception modules. A bottleneck layer is used to lower the input dimensions (i.e., the depth), reducing the number of parameters and the processing costs. A 1D convolution can be applied as a 1D sliding filter that is able to filter out its discriminate region in the time series, speeding up training and improving generalization. The output from the bottleneck feeds three one-dimensional convolution layers with kernel sizes of 10, 20, and 40. In addition, the inception module's inputs pass through the max-pooling layer of size 3 through the bottleneck layer. The outputs of the four convolution layers are sequenced along the depth dimension through the depth concatenation layer. Except for the sequence layer, all layers have the same stride and padding. However, all convolution layers come with 32 filters, and residual connections are used for every third inception module. The network consists of a series of inception modules followed by a batch normalization layer, GlobalAveragePooling1D layer, and a dense layer with a softmax activation function [31,32]. The Inception Time architecture is illustrated in Figure 2.

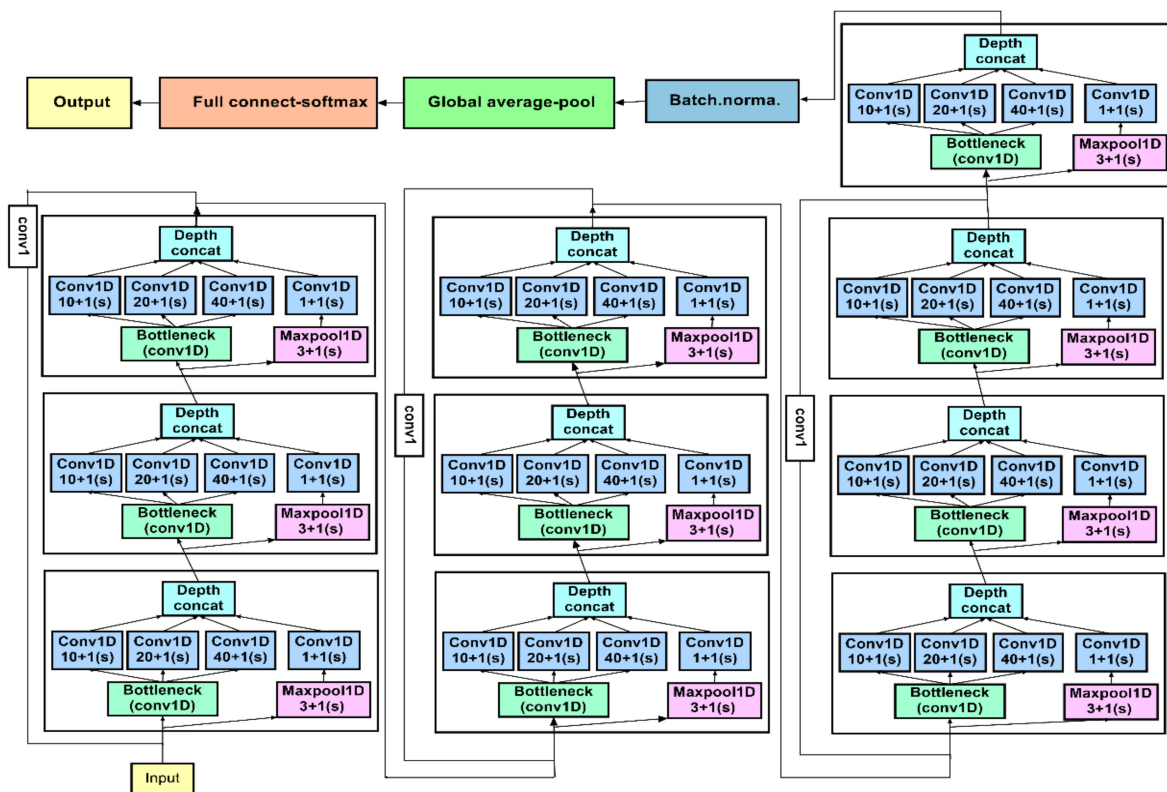


Figure 2. Inception Time architecture.

As shown in Figure 3, the inception module is the most critical component of Inception Time. The first layer acts as a bottleneck, reducing the inputs' dimensionality, the number of parameters, and the computational cost, allowing for faster training and improved generalization. The inception module's second major component is a set of parallel convolutional layers of varying sizes that work on the same input feature map. Figure 3 depicts three unique convolutions with filter sizes of 10, 20, and 40. The third layer is the max-pooling layer, which gives the model the capacity to be invariant to tiny perturbations. The depth concatenation layer is the fourth and final layer; it concatenates the outputs of each independent parallel convolution and the max pooling to produce the current inception module's output multivariate time series.

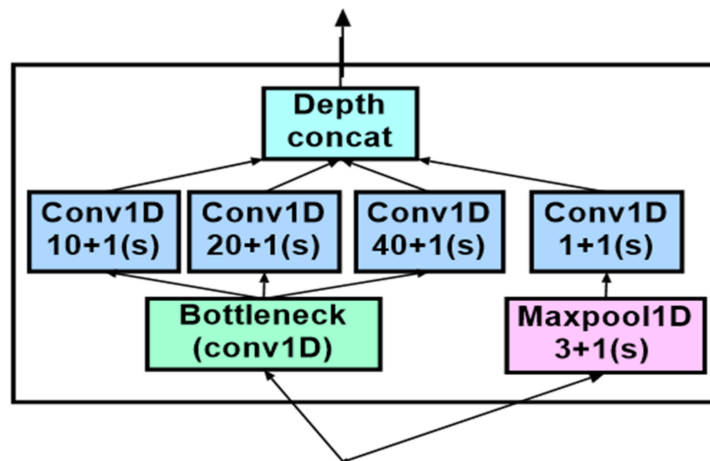


Figure 3. The Inception Time model. The first number in each box represents the kernel size, while the second represents the stride size; (s) indicates the padding type.

It is critical to “learn” the characteristics from all prior windows, regardless of the attack occurrence, to develop a thorough model of the network patterns in the entire dataset. Using a sliding window technique, in which a single packet goes through each window to see if the previous $(T - 1)$ packets have resulted in an attack in the current package, Figure 4 depicts this model. A single window with T packets and n features, with multiple labels for the entire window illustrating the attack in the last packet, and using the deep learning model to determine the attack in the T package, requires learning from the information of the windows’ $(T - 1)$ component packets. Because of the sliding windows, the m packets produce a sum of $(m - T + 1)$ windows [13], as shown in Figure 4.

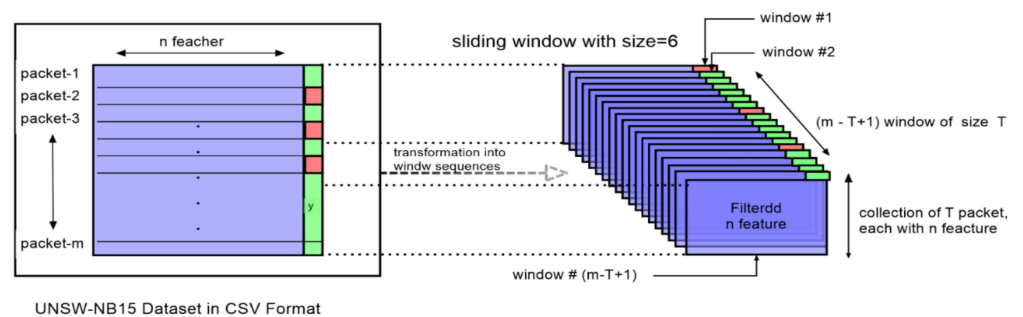


Figure 4. How the input data from the UNSW2015 dataset is turned into window sequences for Inception Time model training using the sliding window technique [13].

Inception Model Description

#Function inception_module

- Input: simple
 - Output: staked Layers
1. If use_bottleneck and input_tensor last_dimmensional > 1:
 - a. input_inception = Conv1D (input_tensor, filters = bottleneck_size, kernel_size = 1, padding = ‘same’, activation = activation, use_bias = False)
 2. Else:
 - a. input_inception = input_tensor
 3. Kernel_size_s = [kernel_size // (2 ** i) for i in range(3)]
 4. Conv_list = []
 5. For i in range (len(kernel_size_s)):

- a. `conv_list.append (Conv1D)`
6. `Max_pool = MaxPool1D (input_tensor, pool_size = 3, strides = stride, padding = 'same')`
7. `Conv = Conv1D (max_pool, filters, kernel_size = 1, padding = 'same', activation`
8. `x = Concatenate (conv_list, axis = 2)`
9. `x = BatchNormalization (x)`
10. `x = Activation (x, activation = 'relu')`

Function shortcut_layer

1. `Shortcut = Conv1D (input_tensor, filters = out_tensor last dimensional, kernel_size = 1, Padding = 'same', use_bias = False)`
2. `Shortcut = BatchNormalization(Shortcut)`
3. `x = Add ([Shortcut, out_tensor])`
4. `x = Activation (X, activation = 'relu')`

Function inception_time_model

1. `Input_layer = Input (input_shape)`
2. `x = input_layer`
3. `Input_res = input_layer`
4. For `d` in range (depth):
 - a. `x = inception_module (x)`
 - b. if `use_residual` and `d % 3 == 2`:
 - i. `x = shortcut_layer (input_res, x)`
 - ii. `input_res = x`
5. `Gap_layer = GlobalAveragePooling1D()(x)`
6. `Output_layer = Dense (gap_layer, number_classes, activation = 'softmax')`
7. `Model = Model (inputs = input_layer, outputs = output_layer)`

5. Design of Experiments

5.1. Datasets

The models in this article were evaluated using the ToN-IoT, Edge-IIoT, and UNSW2015 databases. The ToN-IoT dataset is made up of heterogeneous sources, including Win7, Win10, Network, and IoT. The dataset contains 2,233,921 records of normal and attack data. The training set had 22,339,021 network packet vectors, while the test set included 461,043 vectors and 43 features in CSV format, which can be utilized on any platform. The UNSW-NB15 dataset has over 100 GB of network packets split among 2,540,044 vectors and 42 features in four CSV files. The data include normal vectors and a variety of attack vectors (backdoor exploits, DoS, shellcode, worms, and reconnaissance are examples of fuzzers). The Edge-IIoT dataset contains 14 different types of attacks related to IoT and IIoT protocols, which are divided into five categories (i.e., DoS and DDoS attacks, information gathering, injection attacks, man-in-the-middle attacks, and malware attacks). The Edge-IIoT dataset contains 1,909,671 samples, 15 classes, and 61 different features. Sets of 1,527,736 and 381,935 data for training and testing are available.

These three datasets were divided into 80% training and 20% testing sets, reducing the risk of overfitting and allowing for comprehensive model performance monitoring over the entire dataset. Because the data were balanced, we utilized the class weights. After conducting some experiments to tune the hyperparameters, the batch size was 64, which is suitable for memory that stabilizes the training process without oscillation and converges to general performance. The depth was reduced to 5 or 6 for some data and remained 10 for others; some dataset need a more complex model to converge to a satisfactory F1-score, while some converge with a module depth of 5 or less. However, we tried to keep the residual layer after every three modules. In terms of the number of epochs, 50 is an overestimate that may not be reached, because the model was stopped early if it began to overfit by tracing the validation accuracy. We used the Adam optimizer and a learning scheduler to converge quickly and to find the best learning rate.

5.2. Data Processing

This paper uses features from ToN-IoT, UNSW-NB15, and Edge-IIoTset—three well-known datasets. There are many challenges in these datasets, including behavioural duplicates, missing values, and unnecessary features that affect the models' performance. We employed class weights to balance the data and enhance the classification performance.

The ToN-IoT dataset is displayed in CSV format and includes a sorted column for each attack subclass type, along with the attack's typical behaviour or attack state. It refers to nine different attack types, where the mean is used to fill in all of the NaN values.

In UNSW-NB15, the attack samples are divided into nine classes in a CSV file correctly named "normal" or "attack". We identified samples with missing values, such as NaN, and discarded them. We also populated the NaN class with normal vectors, set the NaN values to zero, and discarded static characteristics that were of no use (e.g., srcip, sport, dstip, dsport).

Regardless of the timing of the attack, it is crucial to "know" the properties of all prior windows when using a sliding window technique. A single packet transmits each window to determine whether earlier packets (T-1) caused an attack on the present packet.

Edge-IIoT is organized into five categories and is in CSV format. We checked samples such as NaN that had missing values and discarded them along with any duplicates. Additionally, we dropped static features with the same value in the whole dataset (e.g., icmp. unused, http. tls_port, dns. qry. type, mqtt. msg_decoded_as).

5.3. Metrics for Evaluation

The models were trained using datasets with varying degrees of training data to assess their accuracy, false positive rate, accuracy, and detection rate, as well as their F1-score. These measurements were constructed using true negative (TN), true positive (TP), false negative (FN), and false positive (FP) data. The improperly classified legitimate and attack vectors were FP and FN, respectively. TP and TN denote the numbers of successfully classified legitimate and attack vectors [33,34].

Accuracy (ACC): The system's capability to categorize attack packets as normal or attack. For all samples, any percentage forecasts are acceptable. Accuracy is mathematically stated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP} \times 100 \quad (1)$$

Precision (Pre): Defines the percentage of genuinely detected attacks versus all packets designated as attacks; arithmetically expressed as follows:

$$\text{Precision} = \frac{TP + TN}{TP + FP} \times 100 \quad (2)$$

F1-score (F1) is theoretically defined as the harmonic average of recall and precision.

$$\text{F1 - Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Recall (Rec): The system's ability to correctly detect attacks when a security breach occurs; often known as the true positive rate, and expressed mathematically as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \times 100 \quad (4)$$

6. Experimental Results

In our tests, DenseNet was solely applied to the ToN-IoT dataset, while Inception Time was applied to the ToN-IoT, UNSW2015, and Edge-IIoT datasets. Inception Time is the new face of research. Developed to reduce computational burden as well as overprocessing while obtaining better performance, it allows features to be extracted at different scales by using different-sized filters. We found that using Inception Time increased the accuracy.

Therefore, we tested its efficacy on several other datasets and decided to focus this work on Inception Time.

6.1. ToN-IoT Dataset

The previously described ToN-IoT dataset contains heterogeneous data from Win7, Win10, Network, and IoT. We evaluated its efficiency using DenseNet and the Inception Time algorithm in a multi-class scenario.

We used a DenseNet with a depth of 121 on the ToN-IoT dataset for evaluation, and divided it into 80% training and 20% testing sets. The experiments were based on the different features of the Win7, Win10, Network, and Win10–Network training models. As shown in Table 6, the results showed that the model achieved good results. The best accuracy was recorded for the Win10–Network model—up to 99.9%—and the lowest accuracy we obtained for Win10 was 97.7%. The results of Win7 and Network were 98.36% and 98.5%, respectively. It should be noted that we used a hyperparameter, a starting learning rate of 0.005, and a batch size of 64. Figure 5 shows the accuracy, loss, recall, precision, and F1-score of the DenseNet algorithm.

Table 6. The values of training and testing, the numbers of features used, and the values of the epochs for the Inception Time and DenseNet algorithms.

Model	Type			
DenseNet	Win7	Win10	Network	Win10–Network
Training	22,693	28,780	17,582,906	859,003
Testing	5674	7195	4,395,726	214,751
Features	132	124	42	166
Epoch of 50	29	50	1	3
Inception time				
Training	22,693	28,780	17,582,906	859,003
Testing	5674	7195	4,395,726	214,751
Features	132	124	42	166
Epoch of 50	32	42	1	5

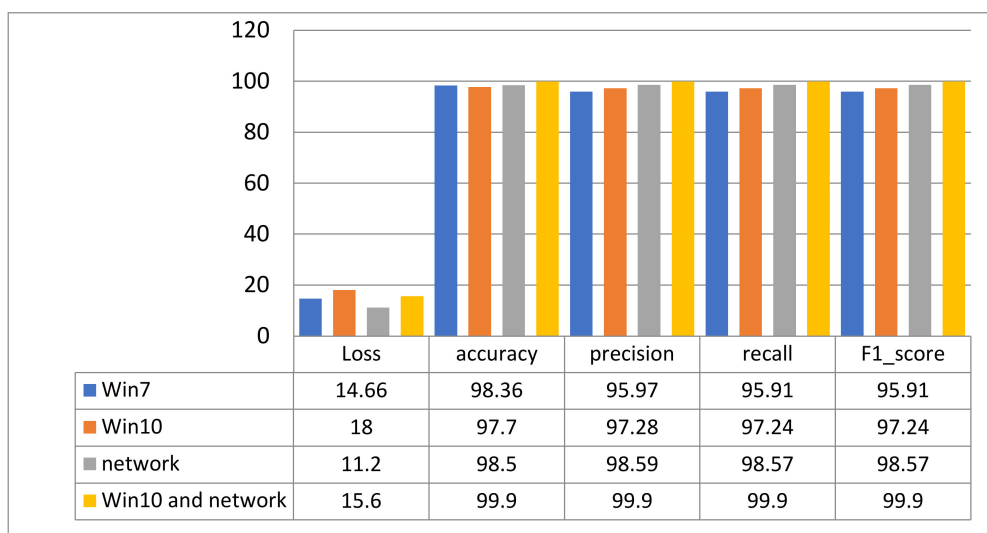


Figure 5. The results of the DenseNet model of the dataset.

Figure 6 shows the achieved training and validation accuracies after 29 Win7 epochs, 50 Win10 epochs, and 5 Win10–Network epochs in the ToN-IoT dataset—which consists of Win7, Win10, Network, and Win10–Network—using the DenseNet algorithm.

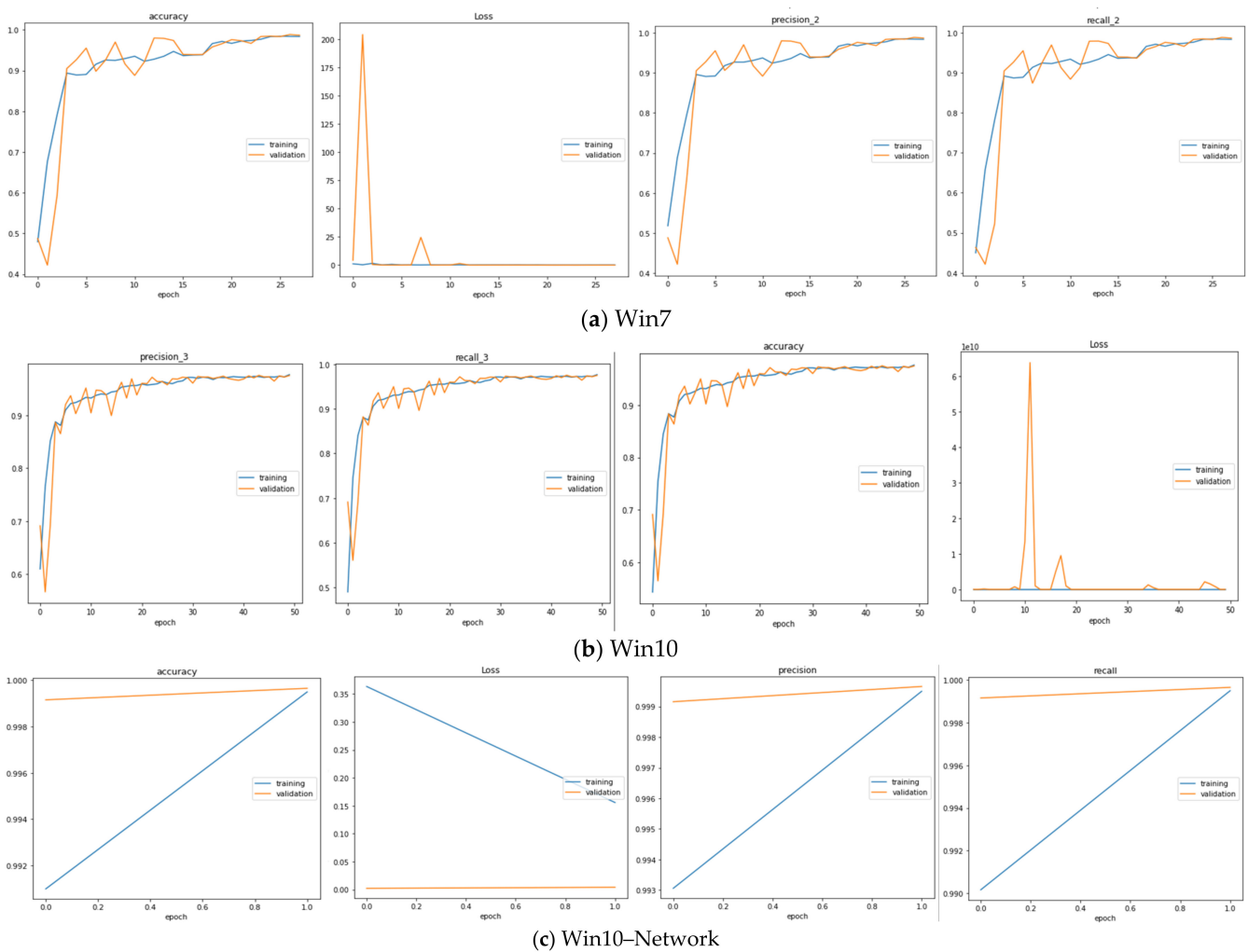


Figure 6. (a–c) The training and validation accuracy, F1-score, loss, recall, and precision after training the DenseNet model on the ToN-IoT dataset.

We also used the Inception Time algorithm to examine the performance of the same ToN-IoT dataset with the same training and testing values and the prior features of Win7, Win10, and Network, as shown in Table 6. All of the results indicated the superiority of the Inception Time model over the DenseNet model, as shown in Figure 7. The best accuracy we obtained was 100% for the Win10–Network model, followed by Network, with an accuracy of 99.65%, while the lowest accuracy we obtained in Win10 was 98.3% and for Win7 it was 99.2%, with a starting learning amount of 0.001 and an end-learning amount of 0.001, while maintaining the same batch size of 64.

Figure 8 shows the achieved training and validation accuracies after 32 Win7 epochs, 42 Win10 epochs, and 5 Win10–Network epochs in the ToN-IoT dataset—which consists of Win7, Win10, Network, and Win10–Network—using the Inception Time algorithm.

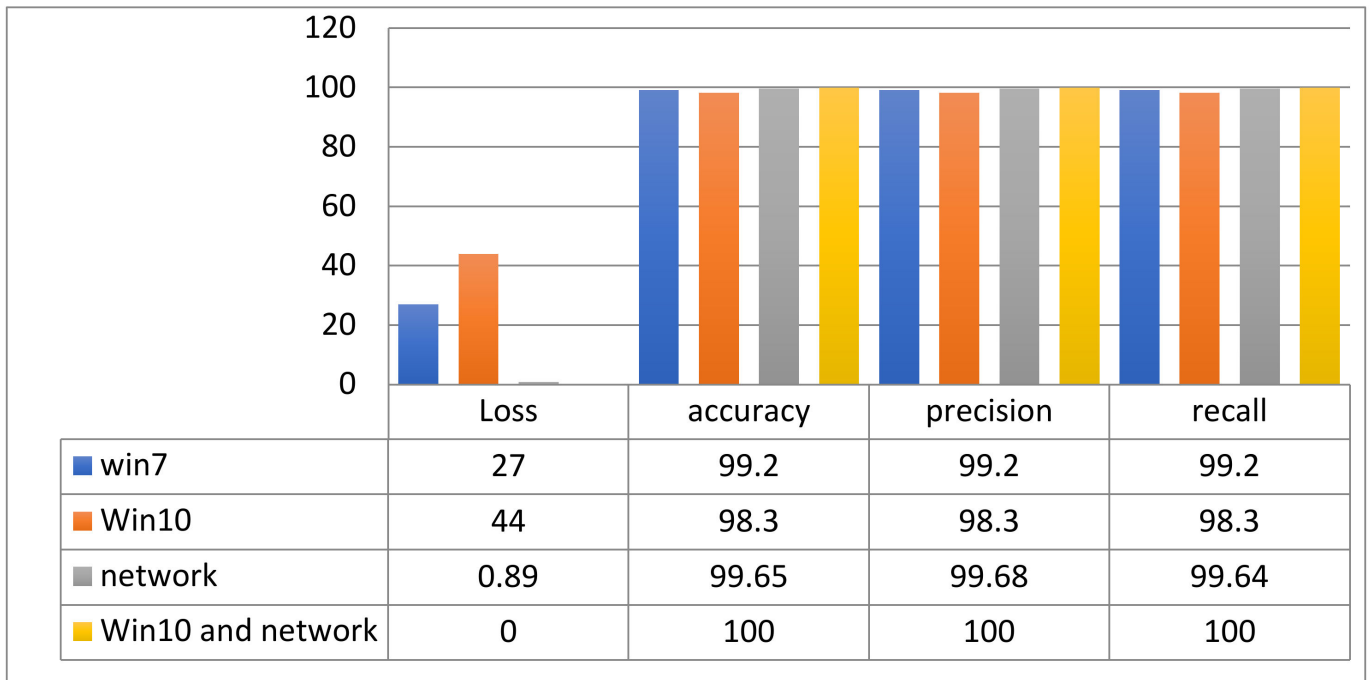


Figure 7. The results (Win7, Win10, Network, Win10–Network) of the Inception Time model of the ToN-IoT dataset.

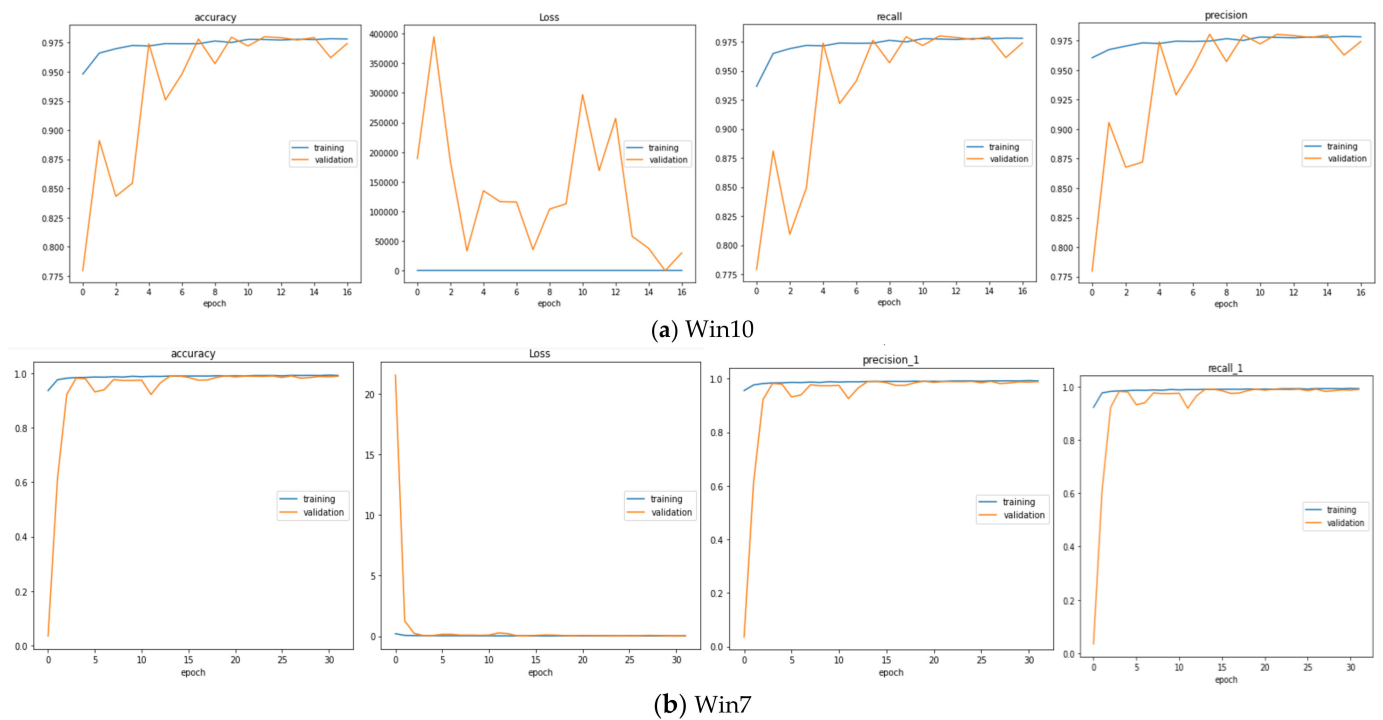
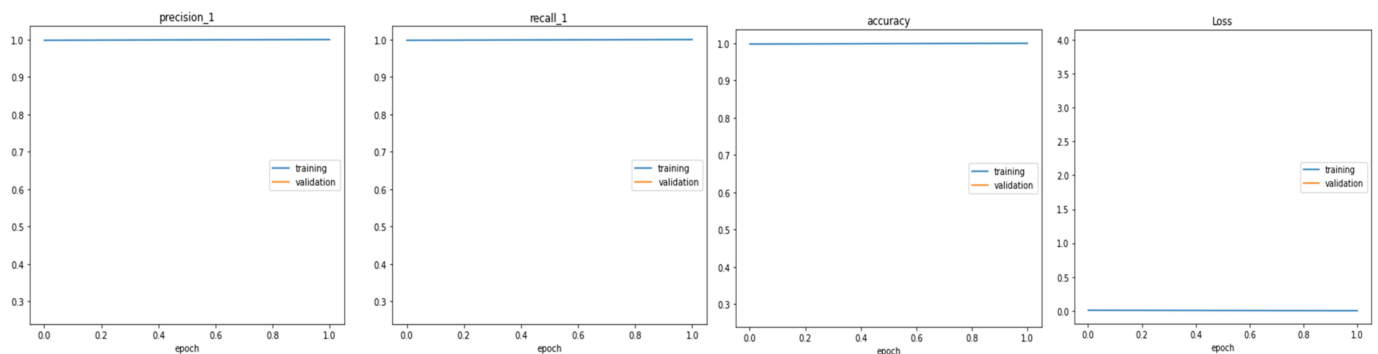


Figure 8. Cont.



(c) Win10-Network

Figure 8. (a–c) The training and validation accuracy, F1-score, loss, recall, and precision on the ToN-IoT dataset after training with the Inception Time model.

Weather, thermostat, motion lighting, Modbus, garage, GPS, and fridge are the seven categories of IoT in the ToN-IoT database. To assess the performance, we used the Inception Time technique in a multi-class scenario with six classes (i.e., DDoS, password, normal, backdoor, scan, and XXS) with all features enabled; the results demonstrated its correctness. We achieved the highest accuracy of 100% for weather, thermostat, and GPS, and the lowest value we received for the fridge was 99%. In contrast, the Modbus, motion lighting, and garage accuracies were 99.9%, 99.5%, and 99.4%, respectively. Figure 9 displays the actual findings for all categories of IoT in the ToN-IoT database using Inception Time.

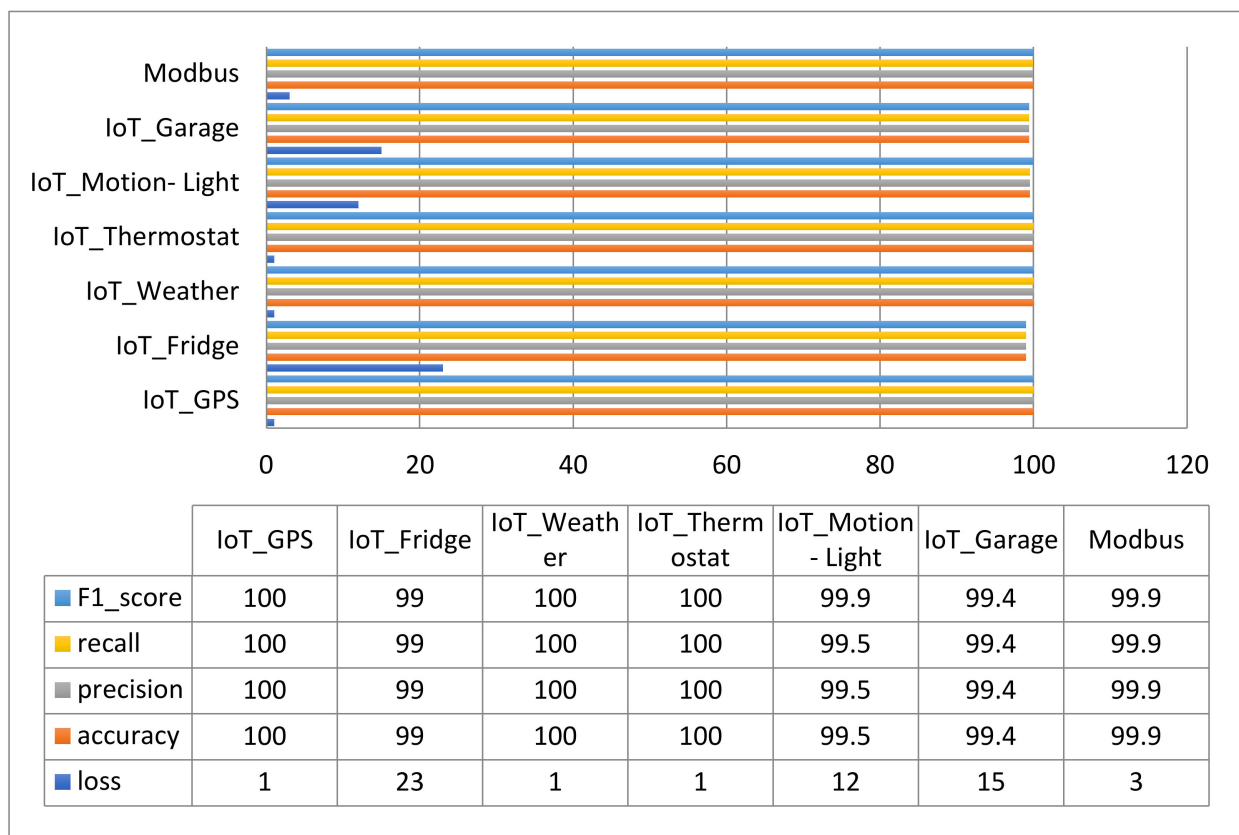


Figure 9. The results of the IoT network in the inception time model of the ToN-IoT dataset.

Experiments on the ToN-IoT database showed that the Inception Time model method outperformed DenseNet. The best accuracy we obtained using DenseNet was 99.9% for

Win10–Network, while the lowest was 97.7% for Win10. However, when using the Inception Time model, the maximum accuracy was 100% in the Win10–network model, while the lowest accuracy was 98.3% in the Win10 model, as shown in Figure 10.

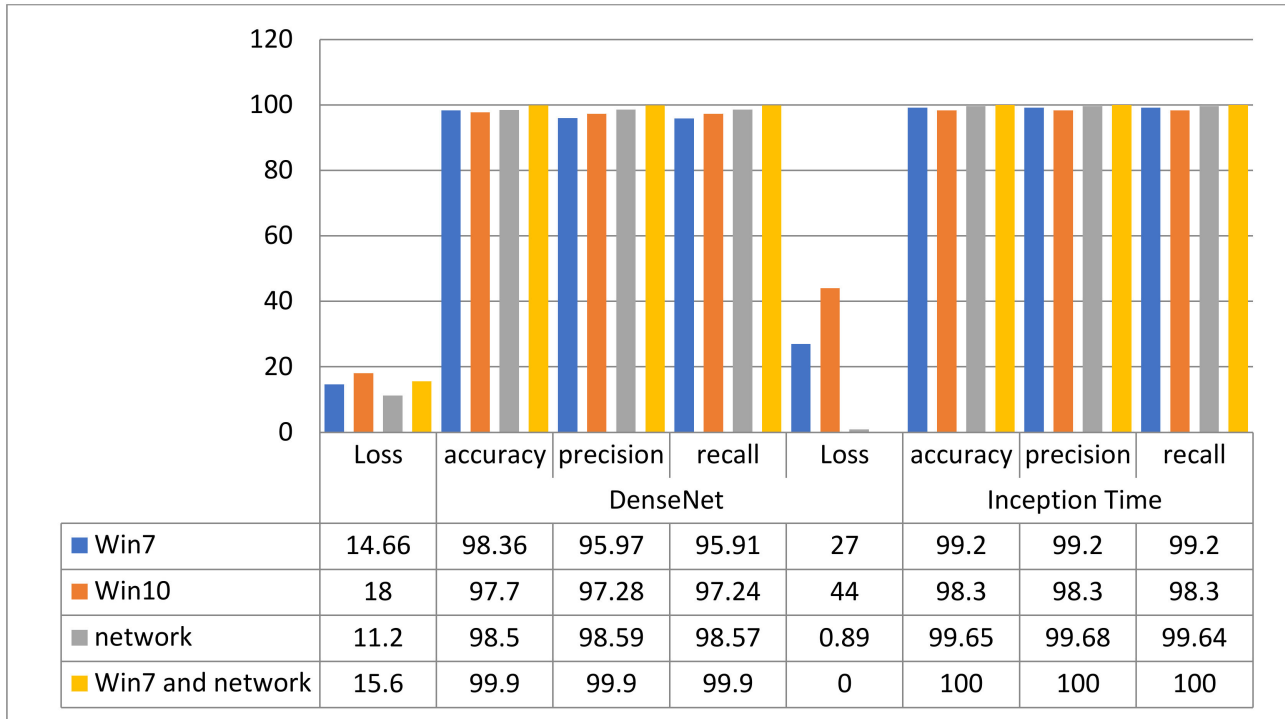


Figure 10. The comparison of results between DenseNet and Inception Time using the ToN-IoT dataset.

Compared to the previous trials in Table 7, the Inception Time model outperformed each (i.e., DT, NB, and XGBoost) in categorizing multicategory attacks. Table 8 further indicates our model’s superiority in classifying multi-class assaults in IoT devices (fridge, garage, GPS, Modbus, motion lighting, thermostat, weather) and utilizing the ToN-IoT dataset.

Table 7. Comparison of the results of previous studies using the ToN-IoT dataset.

Research	Method	Type	Acc	Pre	Rec	F1	Class
Sarhan M., et al. [20]	DT	Network	97.29	-	-	99	Multi
	NB		96.78			98	
Gad A.R., et al. [12]	XGBoost	Network	98.3	98.3	98.3	98.3	Multi
Our proposed method	DenseNet	Win7	98.36	95.97	95.91	95.91	Multi
		Win10	97.87	97.87	97.87	97.87	
		Network	98.57	98.59	98.57	98.57	
		Win10–N	99.95	99.95	99.95	99.95	
	Inception Time	Win7	99.21	99.21	99.21	99.21	Multi
		Win10	98.30	98.30	98.30	98.30	
		Network	99.65	99.68	99.64	-	
		Win10–N	100	100	100	-	

Table 8. Comparison of the results of previous studies using the IoT devices in the ToN-IoT dataset.

Research	Model	KNN						
A. Alsaedi, et al. [34]	Dataset	Fridge	Garage	GPS	Modbus	Motion Lighting	Thermostat	Weather
	Accuracy	0.99	1.00	0.88	0.97	0.54	0.60	0.81
	Precision	0.99	1.00	0.89	0.97	0.34	0.56	0.81
	Recall	0.99	1.00	0.88	0.97	0.59	0.61	0.81
	F1-score	0.99	1.00	0.88	0.97	0.43	0.57	0.81
	LSTM							
	Dataset	Fridge	Garage	GPS	Modbus	Motion Lighting	Thermostat	Weather
	Accuracy	1.00	1.00	0.87	0.68	0.59	0.66	0.82
	Precision	1.00	1.00	0.89	0.46	0.35	0.45	0.82
	Recall	1.00	1.00	0.88	0.68	0.59	0.67	0.81
F1-score	1.00	1.00	0.88	0.55	0.44	0.54	0.80	
RF								
D. Rani, et al. [35]	Dataset	Fridge	Garage	GPS	Modbus	Motion Lighting	Thermostat	Weather
	Accuracy	0.9136	0.9314	0.92		0.9216	0.9532	0.9669
	Precision	0.89	0.90	0.92		0.89	0.92	0.97
	Recall	0.91	0.93	0.93		0.92	0.95	0.97
	F1-score	0.89	0.91	0.92		0.90	0.94	0.96
LGBM								
	Dataset	Fridge	Garage	GPS	Modbus	Motion Lighting	Thermostat	Weather
	Accuracy	0.9135	0.9314	0.94		0.9211	0.9532	0.9680
	Precision	0.98	0.90	0.95		0.89	0.92	0.97
	Recall	0.91	0.93	0.95		0.92	0.95	0.97
	F1-score	0.89	0.91	0.95		0.90	0.94	0.97
Inception Time								
Our proposed method	Dataset	Fridge	Garage	GPS	Modbus	Motion Lighting	Thermostat	Weather
	Accuracy	0.990	0.994	1.00	0.999	0.995	1.00	1.00
	Precision	0.990	0.994	1.00	0.999	0.995	1.00	1.00
	Recall	0.990	0.994	1.00	0.999	0.995	1.00	1.00
F1-score	0.990	0.994	1.00	0.999	0.995	1.00	1.00	

6.2. Edge-IIoT Dataset

This experiment sought to evaluate the performance of the Inception Time model for malware detection using the Edge-IIoT dataset, which consists of 1,527,736 training data, 381,935 test data, and 91 features to ensure the reliability of the evaluation. We used four measures based on multiple classes: accuracy, precision, recall, and F1-score. Our best accuracy was 94.94%, as shown in Figure 11, which shows the percentage of attacks predicted properly for 15 classes of the confusion matrix illustrated in Figure 12, each of these was calculated independently, with a starting learning rate of 0.001, batch size of 64, and 50 epochs. The end-learning rate was 0.001.

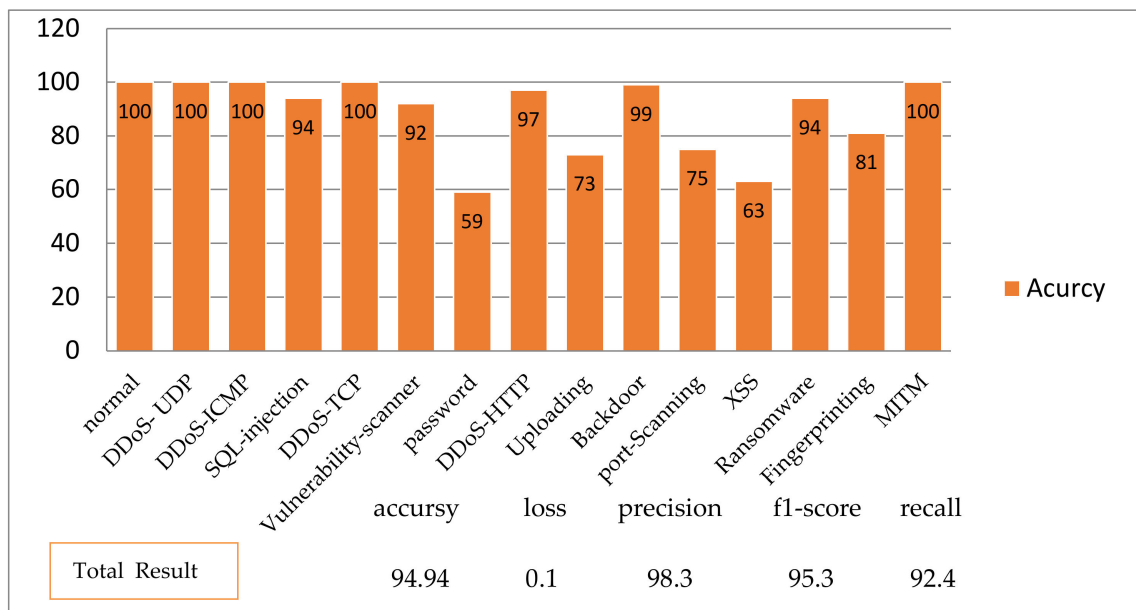


Figure 11. The results of Inception Time and multi-class classification of the Edge-IIoT dataset.

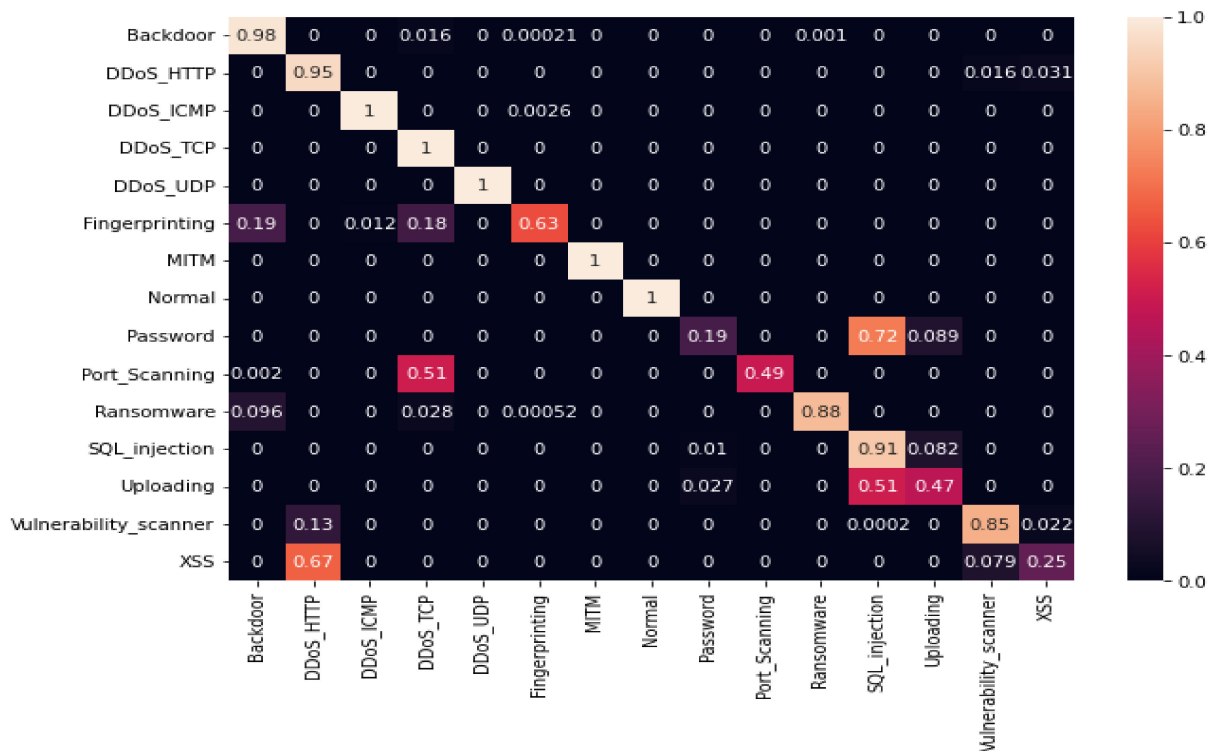


Figure 12. The confusion matrix for 15 classes in the Edge-IIoT-Inception Time validation: DDoS-HTTP, backdoor, DDoS-ICMP, DDoS-TCP, DDoS-UDP, fingerprinting, normal, password, port scanning, SQL-injection, uploading, XSS, vulnerability scanner, ransomware, and MITM; all values between 0.59 and 100.

Figure 13 shows the achieved training and validation accuracies; it could be observed that, after 34 epochs, the Edge-IIoT dataset used the Inception Time algorithm.

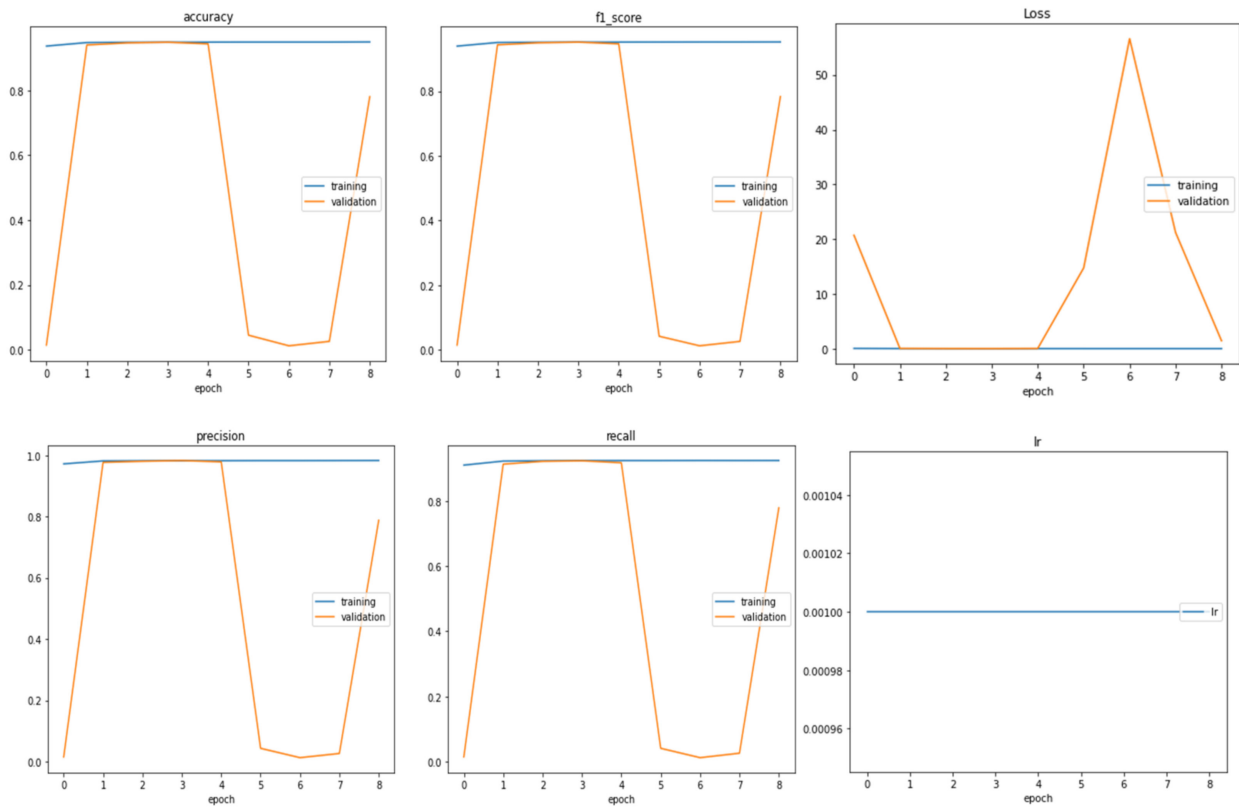


Figure 13. The training and validation accuracy, F1-score, loss, recall, and precision after training the Inception Time model on the Edge-IIoT dataset.

Table 9 compares our model to the previous study, demonstrating that the Inception Time model outperforms RF, SVM, KNN, and DNN in classifying multiple attacks, with the best result being 94.94% on the Edge-IIoT dataset.

Table 9. Comparison of the results with previous studies using the Edge-IIoT dataset.

Research	Class	Model	Accuracy
M. A. Ferrag, et al. [22]	15	RF	80.83
		SVM	77.61
		KNN	79.18
		DNN	94.67
Our proposed method	15	Inception Time	94.94

6.3. UNSW-NB15 Dataset

We evaluated the performance of the Inception Time model on the UNSW-NB15 database to detect cyber-attacks. The training and test sets consisted of 2,032,037 and 508,010 data, respectively, in the case of 10 classes, and the feature number was 43. The Inception Time model was the subject of the first experiment, which yielded an accuracy of 98.4%. The percentage of multicategory attacks is shown in Figure 14. The correlation matrix in Figure 15 was used to calculate each category separately.

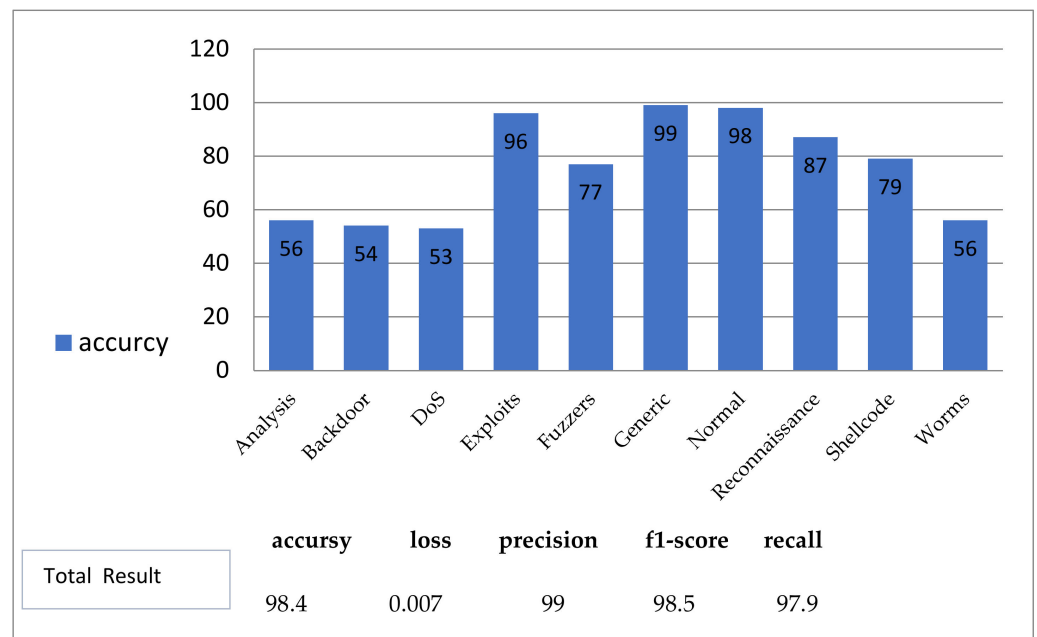


Figure 14. The Inception Time and multi-class results for the UNSW-NB15 dataset.

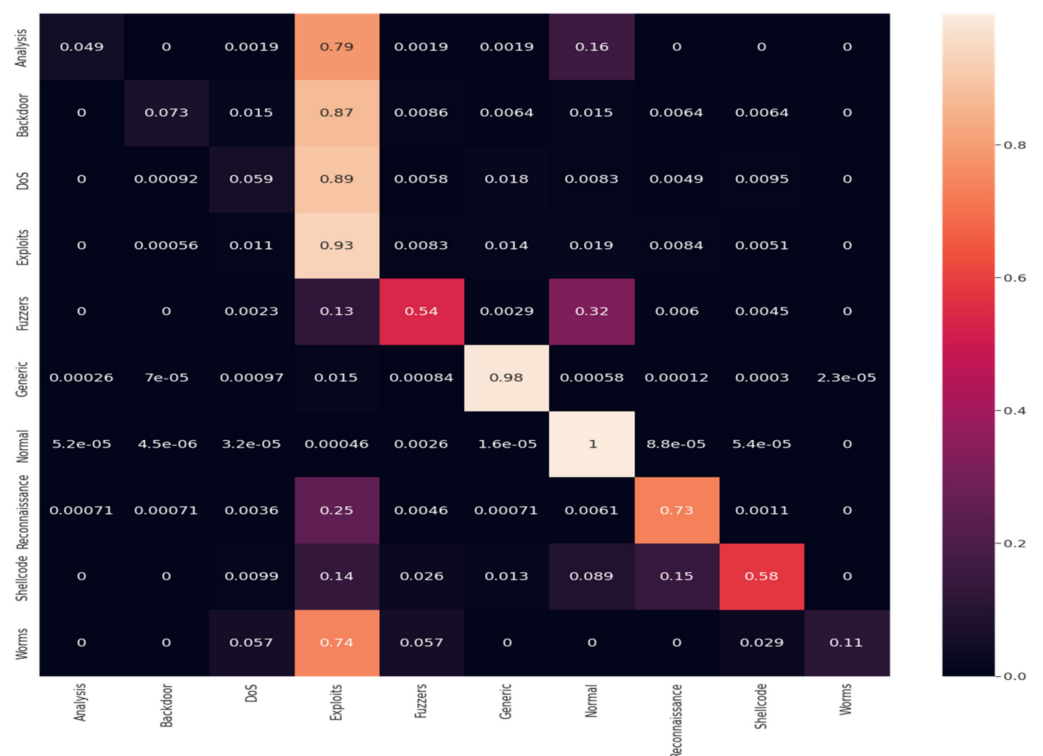


Figure 15. The confusion matrix for 10 classes in the Inception Time validation using the UNSW-NB15 dataset: normal, exploits, generic, fuzzers, reconnaissance, DoS, backdoor, analysis, shellcode, and worms; all values between 0.53 and 99.

Figure 16 shows the achieved training and validation accuracies after 36 epochs in the UNSW-NB15 dataset using the Inception Time algorithm.

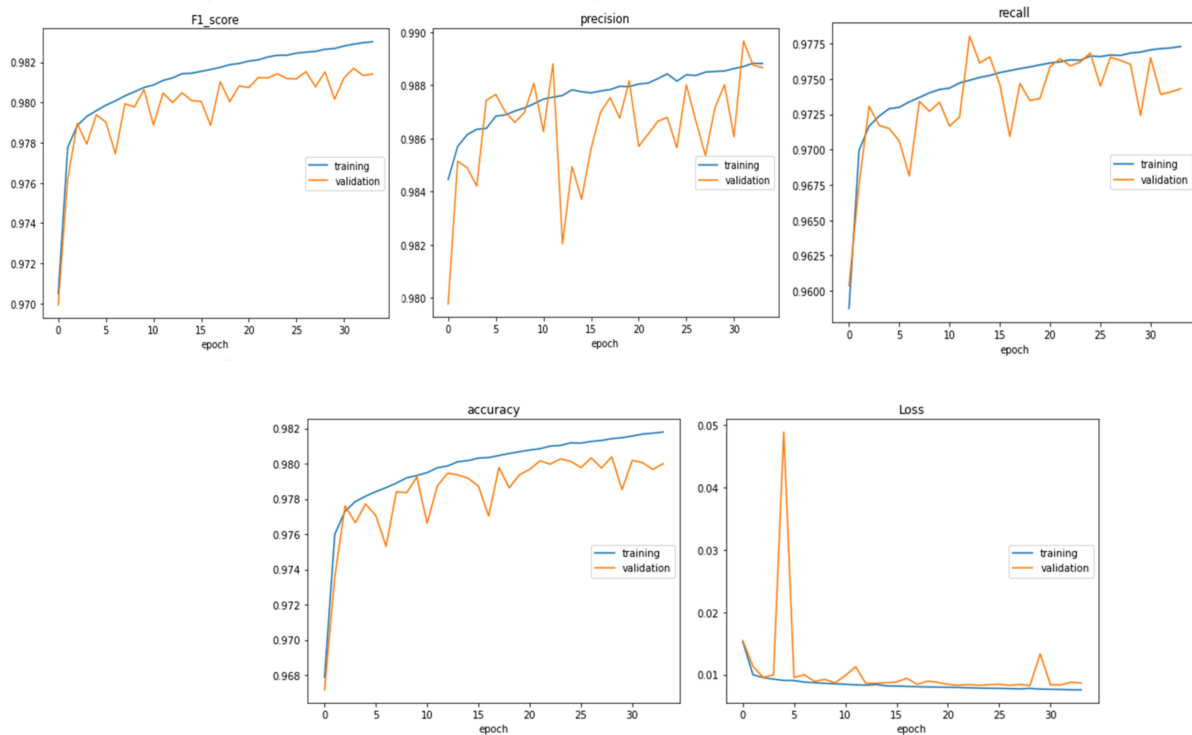


Figure 16. The training and validation accuracy, F1-score, loss, recall, and precision after training the Inception Time model using the UNSW-NB15 dataset.

The second experiment used Inception Time with a window size of six. The UNSW-NB15 dataset’s network packets were used in the preprocessing stage to extract relevant features and turn individual packets into window sequences, where $X_T = (t - window_size, t - 3 + t - 2 + t - 1 + T)$. In addition, $y_t = y$ of the current window at time t . We evaluated its efficiency in multi-class scenarios with 43 features. The results showed a slight increase in accuracy to 98.6%. Figure 17 shows the percentage of multicategory attacks, and each category was calculated separately from the correlation matrix shown in Figure 18, with a starting learning rate of 0.005, batch size of 2048, and 50 epochs. The end-learning rate was 0.0015.

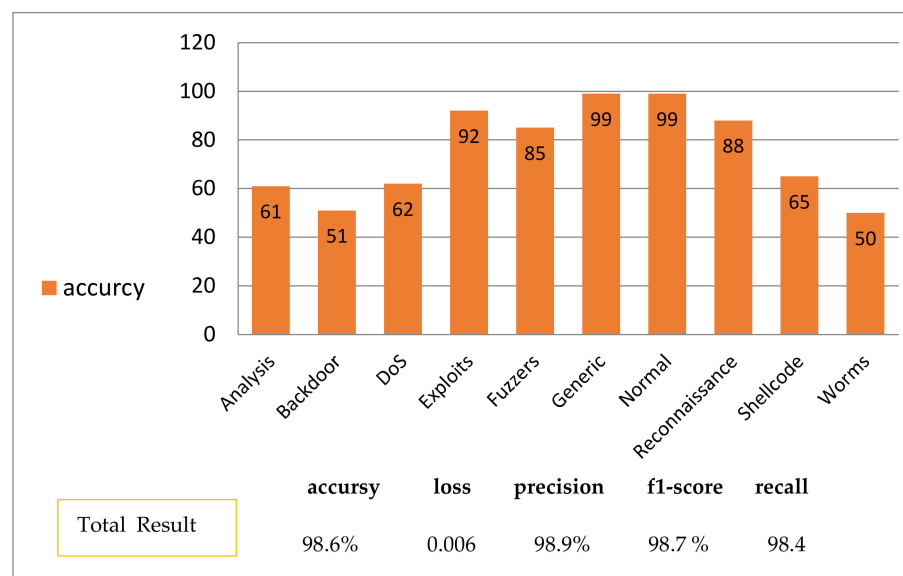


Figure 17. The Inception Time and multi-class results of the UNSW-NB15 dataset with a window size of six.

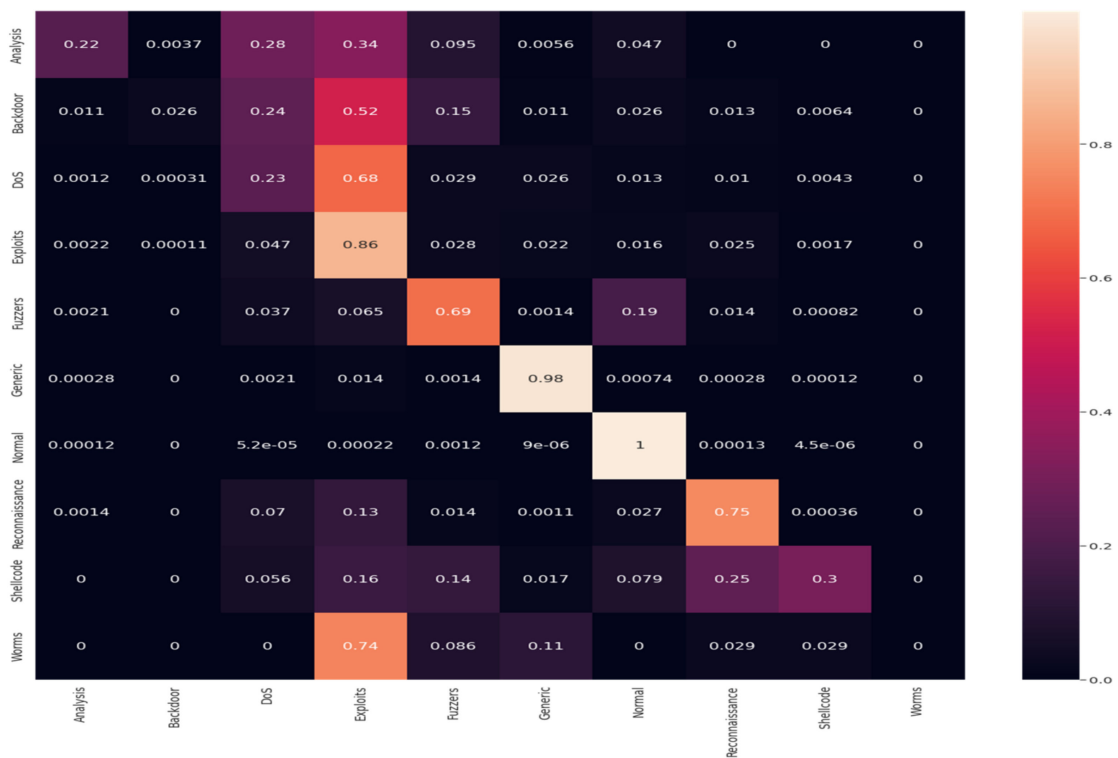


Figure 18. The confusion matrix for 10 classes in the Inception Time validation with a window size of six using the UNSW-NB15 dataset: normal, fuzzers, generic, exploits, reconnaissance, DoS, analysis, backdoor, worms, and shellcode; all values between 0.50 and 99.

Figure 19 shows the achieved training and validation accuracies after 36 epochs in the UNSW-NB15 dataset using the Inception Time algorithm with window size.

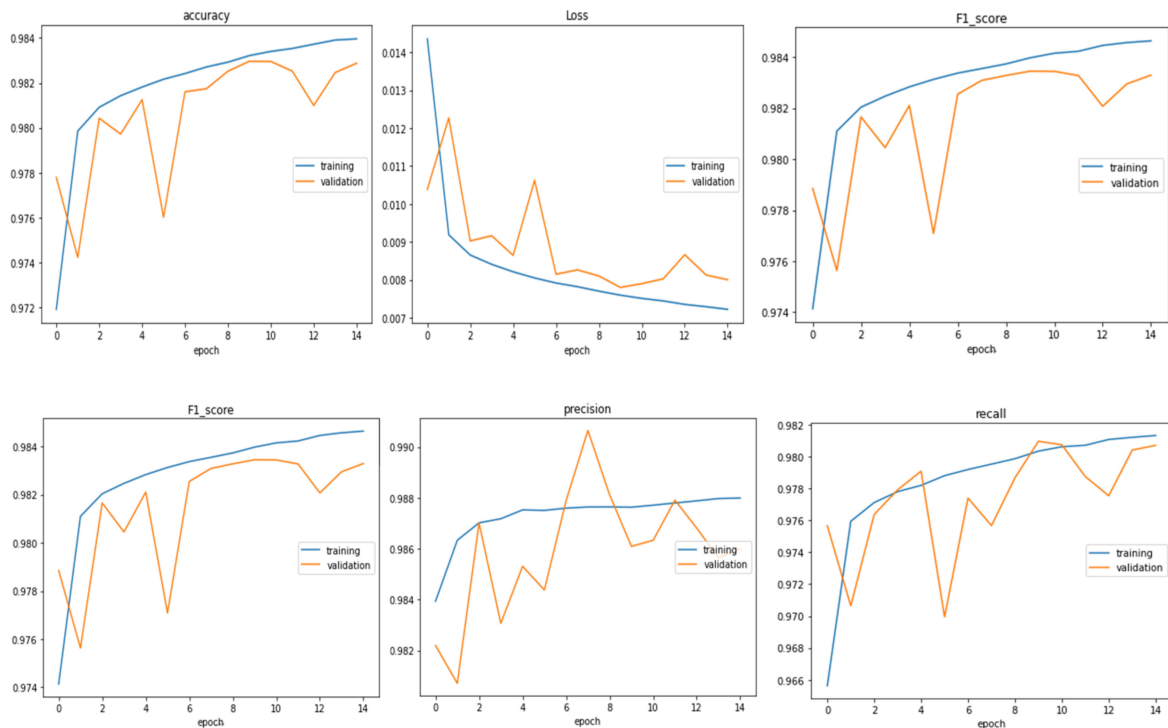


Figure 19. The training and validation accuracy, F1-score, loss, recall, and precision after training the Inception Time model with a window size of six using the UNSW-NB15 dataset.

Table 10 compares the Inception Time models in detecting the initial cyber-attacks: the first using 1D inputs, and the second with a two-dimensional time-series input utilizing the UNSW-NB15 database and size-six sliding window. We observed the superiority of sliding window technology in some types of attacks.

Table 10. Classification report comparison between Inception Time and inception time with six windows using the UNSW-NB15 dataset.

Type Class	Inception Time			Inception Time–Window Size Six			Support
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
Analysis	0.42	0.05	0.09	0.52	0.22	0.31	535
Backdoor	0.69	0.07	0.13	0.75	0.03	0.05	466
DoS	0.51	0.06	0.11	0.39	0.23	0.29	3271
Exploits	0.58	0.93	0.72	0.65	0.86	0.74	8905
Fuzzers	0.67	0.54	0.60	0.74	0.69	0.72	4849
Generic	0.99	0.98	0.99	0.99	0.98	0.99	43096
Normal	1.00	1.00	1.00	1.00	1.00	1.00	443,753
Reconnaissance	0.91	0.73	0.81	0.82	0.75	0.78	2798
Shellcode	0.55	0.58	0.57	0.67	0.30	0.42	302
Worms	0.80	0.11	0.20	0.00	0.00	0.00	35
Accuracy	0.984			0.986			
Weighted average	0.98			0.98			

Table 11 compares the Inception Time models to the previous study in detecting and classifying multiple attacks using the UNSW-NB15 database. The best result was 98.6 % using sliding window technology.

Table 11. Comparison of the results with previous studies using the UNSW-NB15 dataset.

Research	Method	Acc	Pre	Rec	F1	Class	Feature
P. Wu, et al. [19]	Densely-ResNet	73.93	80.94	96.68	88.11	Multi	
A. R. Gad, et al. [12]	RF	95.43	0.96	0.97	0.97	Multi	42
	DT	94.20	0.93	0.98	0.96		
R. A. Khamis, et al. [36]	ANN	0.97	0.96	1.00	-	Multi	5
	CNN	0.96	0.95	1.00			
Y. Yin, et al. [13]	MLP	84.24	83.60	84.24	82.85	Multi	23
V. Kanimozhi, et al. [37]	RF-DT	89	0.99	0.85	0.91	Multi	4
S. M. Kasongo, et al. [38]	ANN	75.62	79.92	75.61	76.58	Multi	42
	KNN	70.09	75.79	70.21	72.03		
Our proposed method	Inception	98.4	99.0	97.9	98.5	Multi	43
	Inception6w	98.6	98.9	98.4	98.7		

7. Conclusions

In this paper, work was carried out to evaluate the performance of each of the selected DenseNet and Inception Time models for detecting cyber-attacks, as they were tested on three recent datasets (ToN-IoT, Edge-IIoT, and UNSW-NB15). The results were compared based on accuracy, recall, precision, and F1-score. The results showed the superiority of the Inception Time approach in the ToN-IoT database. The highest accuracy that we obtained with all of the features of the DenseNet approach was 99.9% in the Win10–Network model. There was an increase in accuracy when using the Inception Time approach, with a resolution of 100% for the Win10–Network model based on multiple classes. Inception Time is playing an increasingly important role and is becoming a prominent direction for research. Therefore, we focused our work on this model and implemented it on three datasets.

Moreover, using the Edge-IIoT database with Inception Time with a rating of 15 classes and 61 features, we obtained the highest result, with an accuracy of 94.94%. As for using the UNSW-NB15 database with Inception Time and 43 features for 10 classes, we achieved

an accuracy of 98.4%. The results improved when using the sliding window approach, reaching an accuracy of 98.6%. This is one of the contributions of this paper. One of the challenges we faced in our experiments was memory, so we used class weights instead of smote, where our experiments were within the limited CPU resources of memory, usage, and test time.

Author Contributions: Conceptualisation, I.T. and B.M.E.; formal analysis, I.T. and S.E.-R.; funding acquisition, I.T.; investigation, E.-S.M.E.-H., B.M.E. and S.E.-R.; methodology, I.T. and B.M.E.; supervision, E.-S.M.E.-H., B.M.E. and S.E.-R.; validation, E.-S.M.E.-H., B.M.E. and S.E.-R.; writing—original draft, I.T.; writing—review and editing, I.T., S.E.-R., B.M.E. and E.-S.M.E.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The used dataset in this work (UNSW-NB15 and TON_IoT Datasets) is available Online. <https://research.unsw.edu.au/projects/toniot-datasets> (accessed on 3 April 2022). (EdgeIoTset) Available online: <https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot> (accessed on 8 May 2022).

Acknowledgments: We would to thank Eng. Mohamed Ahmed for his great efforts in the coding of this research, mohamed.ahm.cs@gmail.com.

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

References

1. Jang-Jaccard, J.; Nepal, S. A survey of emerging threats in cybersecurity. *J. Comput. Syst. Sci.* **2014**, *80*, 973–993. [CrossRef]
2. Drew, J.; Moore, T.; Hahsler, M. Polymorphic Malware Detection Using Sequence Classification Methods. In Proceedings of the 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 22–26 May 2016; pp. 81–87. [CrossRef]
3. Canfora, G.; Mercaldo, F.; Visaggio, C.A.; Di Notta, P. Metamorphic Malware Detection Using Code Metrics. *Inf. Secur. J. A Glob. Perspect.* **2014**, *23*, 57–67. [CrossRef]
4. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. 2011__Malware Images, Visualization and Automatic. 2011. Available online: https://vision.ece.ucsb.edu/sites/vision.ece.ucsb.edu/files/publications/nataraj_vizsec_2011_paper.pdf (accessed on 6 April 2022).
5. Kang, H.; Jang, J.-W.; Mohaisen, A.; Kim, H.K. Detecting and Classifying Android Malware Using Static Analysis along with Creator Information. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 479174. [CrossRef]
6. Han, W.; Xue, J.; Wang, Y.; Huang, L.; Kong, Z.; Mao, L. MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. *Comput. Secur.* **2019**, *83*, 208–233. [CrossRef]
7. Zhong, W.; Gu, F. A multi-level deep learning system for malware detection. *Expert Syst. Appl.* **2019**, *133*, 151–162. [CrossRef]
8. Agarap, A.F. Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach Using Support Vector Machine (SVM) for Malware Classification, No. 1. 2017. Available online: <http://arxiv.org/abs/1801.00318> (accessed on 13 April 2022).
9. Zhang, J.; Qin, Z.; Yin, H.; Ou, L.; Zhang, K. A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding. *Comput. Secur.* **2019**, *84*, 376–392. [CrossRef]
10. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
11. Fong, R.C.; Vedaldi, A. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *arXiv* **2017**, arXiv:1704.03296v3.
12. Gad, A.R.; Nashat, A.A.; Barkat, T.M. Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset. *IEEE Access* **2021**, *9*, 142206–142217. [CrossRef]
13. Singh, P.; Jishnu Jaykumar, P.; Pankaj, A.; Mitra, R. Edge-Detect: Edge-Centric Network Intrusion Detection using Deep Neural Network. In Proceedings of the 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2021; pp. 1–6. [CrossRef]
14. Kumar, P.; Tripathi, R.; Gupta, G.P. P2IDF: A Privacy-Preserving based Intrusion Detection Framework for Soft-ware Defined Internet of Things-Fog (SDIoT-Fog). *ACM Int. Conf. Proc. Ser.* **2021**, *2021*, 37–42. [CrossRef]
15. Kumar, P.; Gupta, G.P.; Tripathi, R. TP2SF: A Trustworthy Privacy-Preserving Secured Framework for sustainable smart cities by leveraging blockchain and machine learning. *J. Syst. Arch.* **2021**, *115*, 101954. [CrossRef]
16. Aleesa, A.M.; Younis, M.; Mohammed, A.A.; Sahar, N.M. Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques. *J. Eng. Sci. Technol.* **2021**, *16*, 711–727.

17. Yin, Y.; Jang-Jaccard, J.; Xu, W.; Singh, A.; Zhu, J.; Sabrina, F.; Kwak, J. IGRF-RFE: A Hybrid Feature Selection Method for MLP-Based Network Intrusion Detection on UNSW-NB15 Dataset. 2022. Available online: <http://arxiv.org/abs/2203.16365> (accessed on 19 May 2022).
18. Kumar, P.; Gupta, G.P.; Tripathi, R. A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 9555–9572. [[CrossRef](#)]
19. Wu, P.; Moustafa, N.; Yang, S.; Guo, H. Densely Connected Residual Network for Attack Recognition. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; pp. 233–242. [[CrossRef](#)]
20. Sarhan, M.; Layeghy, S.; Portmann, M. Feature Analysis for ML-based IIoT Intrusion Detection. 2021. Available online: <http://arxiv.org/abs/2108.12732> (accessed on 19 May 2022).
21. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustain. Cities Soc.* **2021**, *72*, 102994. [[CrossRef](#)]
22. Ferrag, M.A.; Friha, O.; Hamouda, D.; Maglaras, L.; Janicke, H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. *IEEE Access* **2022**, *10*, 40281–40306. [[CrossRef](#)]
23. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaria, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. *Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions*; Springer International Publishing: Cham, Switzerland, 2021; Volume 8. [[CrossRef](#)]
24. Enkvetchakul, P.; Surinta, O. Effective Data Augmentation and Training Techniques for Improving Deep Learning in Plant Leaf Disease Recognition. *Appl. Sci. Eng. Prog.* **2022**, *15*, 3810. [[CrossRef](#)]
25. Moustafa, N. ToN_IoT and unsw15 Datasets. Available online: <https://research.unsw.edu.au/projects/toniot-datasets> (accessed on 3 April 2022).
26. Ferrag, M.A. EdgeIIoTset. Available online: <https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iiot-iiot> (accessed on 8 May 2022).
27. Ferrag, M.A.; Friha, O.; Maglaras, L.; Janicke, H.; Shu, L. Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis. *IEEE Access* **2021**, *9*, 138509–138542. [[CrossRef](#)]
28. Bagui, S.; Walauski, M.; Derush, R.; Praviset, H.; Boucugnani, S. Spark Configurations to Optimize Decision Tree Classification on UNSW-NB15. *Big Data Cogn. Comput.* **2022**, *6*, 38. [[CrossRef](#)]
29. Huang, G.; Liu, Z.; Pleiss, G.; Van Der Maaten, L.; Weinberger, K. Convolutional Networks with Dense Connectivity. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**. [[CrossRef](#)]
30. Ji, Q.; Huang, J.; He, W.; Sun, Y. Optimized Deep Convolutional Neural Networks for Identification of Macular Diseases from Optical Coherence Tomography Images. *Algorithms* **2019**, *12*, 51. [[CrossRef](#)]
31. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.-A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [[CrossRef](#)]
32. Fawaz, H.I.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.-A.; Petitjean, F. InceptionTime: Finding AlexNet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [[CrossRef](#)]
33. Dunn, C.; Moustafa, N.; Turnbull, B. Robustness Evaluations of Sustainable Machine Learning Models Against Data Poisoning Attacks in the Internet of Things. *Sustainability* **2020**, *12*, 6434. [[CrossRef](#)]
34. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 165130–165150. [[CrossRef](#)]
35. Rani, D.; Gill, N.S.; Gulia, P.; Chatterjee, J.M. An Ensemble-Based Multi-class Classifier for Intrusion Detection Using Internet of Things. *Comput. Intell. Neurosci.* **2022**, *2022*, 1668676. [[CrossRef](#)]
36. Khamis, R.A.; Matrawy, A. Evaluation of Adversarial Training on Different Types of Neural Networks in Deep Learning-based IDSs. In Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 20–22 October 2020. [[CrossRef](#)]
37. Kanimozhi, V.; Jacob, P. UNSW-NB15 dataset feature selection and network intrusion detection using deep learning. *Int. J. Recent Technol. Eng.* **2019**, *7*, 443–446.
38. Kasongo, S.M.; Sun, Y. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *J. Big Data* **2020**, *7*, 105. [[CrossRef](#)]