applied sciences

Article

A Cloud Intrusion Detection Systems Based on DNN Using Backpropagation and PSO on the CSE-CIC-IDS2018 Dataset

Saud Alzughaibi * and Salim El Khediri *

Department of Information Technology, College of Computer, Qassim University, Buraydah 52571, Saudi Arabia
* Correspondence: 411116202@qu.edu.sa (S.A); s.elkhediri@qu.edu.sa (S.E.K)

Abstract: Cloud computing (CC) is becoming an essential technology worldwide. This approach represents a revolution in data storage and collaborative services. Nevertheless, security issues have grown with the move to CC, including intrusion detection systems (IDSs). Intruders have developed advanced tools that trick the traditional IDS. This study attempts to contribute toward solving this problem and reducing its harmful effects by boosting IDS performance and efficiency in a cloud environment. We build two models based on deep neural networks (DNNs) for this study: the first model is built on a multi-layer perceptron (MLP) with backpropagation (BP), and the other is trained by MLP with particle swarm optimization (PSO). We use these models to deal with binary and multi-class classification on the updated cybersecurity CSE-CIC-IDS2018 dataset. This study aims to improve the accuracy of detecting intrusion attacks for IDSs in a cloud environment and to enhance other performance metrics. In this study, we document all aspects of our experiments in depth. The results show that the best accuracy obtained for binary classification was 98.97% and that for multi-class classification was 98.41%. Furthermore, the results are compared with those from the related literature.

Keywords: cloud computing; artificial neural networks; deep learning; intrusion detection system; particle swarm optimization; backpropagation; multi-layer perceptron; CSE-CIC-IDS2018

check for updates

Citation: Alzughaibi, S.; El khediri, S. A Cloud Intrusion Detection Systems Based on DNN Using Backpropagation and PSO on the CSE-CIC-IDS2018 Dataset. *Appl. Sci.* **2023**, *13*, 2276. https://doi.org/10.3390/app13042276

Academic Editors: Habib Hamam, Ateeq Ur Rehman and Mohamed Tahar Ben Othman

Received: 19 December 2022
Revised: 2 February 2023
Accepted: 8 February 2023
Published: 10 February 2023

1. Introduction

Over the last decade, we have seen considerable advances in CC. Today, CC services have rich features, such as resource pooling and large networks. Cloud providers offer a platform, services, applications, and infrastructure to the cloud's users. The concept of a decentralized cloud is changing the face of the internet and decreasing the need for hardware and software. On the user side, the only requirement is being able to navigate the user interface of the CC system.

Today, cloud technology is rapidly growing with a considerably increasing number of users, cloud service providers, and threats that target users and providers. As a result, the uploading of sensitive data to cloud storage by users poses many security issues regarding accessibility, availability, confidentiality, and integrity. Moreover, without perfect protection, cloud services' uninterrupted provision leads to a risk of intrusion [1].

The cloud environment is vulnerable to intrusion attacks because of its distributed nature. An IDS can be used to ensure the security of the CC by automatically checking the logs, network flow, and configurations. Nevertheless, traditional IDSs that are host- or network-based are not practical for CC since these IDSs are unable to detect hidden attack paths. This makes traditional IDSs inconvenient for CC because they are not designed with the specific context of CC to provide the necessary protection in such an environment. As a result, using traditional IDS techniques in the cloud environment leads to problems that do not exist in the conventional CC [2].

The two main methods of IDS based on detection are misuse (signature) and anomaly. In anomaly, a profile is created by recording normal behaviors, and then anomalies are determined from the regular pattern that can be marked as intrusions. Signature (misuse) detection is achieved with a pattern relating to known attacks or signatures of vulnerabilities. When the IDS detects malicious code matching these signatures, it raises the alarm and marks this code as an intrusion [2].

The vulnerabilities of IDS in the cloud environment can be solved using multiple evolutionary computing and programming techniques. Therefore, current strategies are geared toward the use of a combination of several methods together in one system to build robust and efficient IDSs [3].

Currently, researchers in this field are experimenting with novel methods using metaheuristic algorithms, which attempt to combine the benefits of these algorithms and artificial neural networks (ANNs) to mitigate any significant downside of IDSs [4].

Over the past few years, the applications of DNNs and metaheuristic optimization algorithms have become quite effective. Furthermore, there are different methods for using optimization algorithms in the IDSs. One of these methods uses the PSO algorithm, which simulates self-learning and social organization in groups of animals, such as birds and fish while finding food [5].

BP is a widely used algorithm for training MLP feedforward neural networks in order to adjust the model's weights and biases. Due to its capacity for learning and flexibility, it has been effectively used in many applications [6]. Furthermore, the PSO algorithm has a powerful capacity for exploitation as well as a quick speed of convergence [7]. Furthermore, using the PSO with neural networks gives better outcomes for IDSs [1].

This study attempts to contribute the previous efforts of researchers in this field using different strategies and an updated dataset to improve the performance of IDSs. The contributions of this study are summarized as follows:

- We performed a comprehensive empirical study on IDSs using MLP-BP and MLP-PSO techniques to enhance performance metric scores and determine which model is more powerful in the cloud environment.
- We tried to provide a general analysis of the CSE-CIC-IDS-2018 dataset, and we used it to validate the proposed models for both binary and multi-class classification.
- We tried to address the shortcomings in the literature and avoid them in our experiments, as well as comparing our results against them.
- We used several evaluation metrics to provide more detailed analysis and a full view of the proposed models' performance.
- In the final analysis, our results show an improvement in performance metric scores, as shown in Section 5.

The rest of this paper is organized as follows: Section 2 discusses related works. Section 3 presents the necessary background information. In Section 4, we show the proposed methods and experimental design. Section 5 presents the results. Section 6 discusses our experimental results and compares them with those in the related literature. Finally, Section 7 concludes this study, followed by future research directions.

2. Related Works

In this section, we will discuss the literary works related to this study, especially those based on DNNs for IDSs, which use the CSE-CIC-IDS2018 dataset. Several researchers have used this approach, with one goal: making the IDSs in the CC more efficient and robust by increasing the model scores (accuracy, precision, etc.).

Tables 1 and 2 show a list of these studies' aims, methodologies, environments, and best performance scores. Although all these studies used the CSE-CIC-IDS2018 dataset, each has its own implementation strategy for the proposed IDSs; the features selection process may differ from study to study, in addition to the number of features and samples, and approaches to data preprocessing and splitting. Furthermore, on the level of the DNN architecture, the tuning of the hyperparameters, the number of layers and neurons, and the activation and optimization functions may vary.

Table 1. Summary of related works.

Author	Year	Aim	Method	Environment
[8]	2020	To understand the nature of an attack	DNN Autoencoder	Python, Keras, Tensorflow
[9]	2020	To tackle the class imbalance problem	DNN	Scikit-learn, Tensorflow, Google Colab
[10]	2020	To identify the capability of IDS models	DNN	Several PCs (using GPU on some PCs)
[11]	2021	To improve detection performance	DNN Autoencoder	Python, Keras, Tensorflow-GPU
[12]	2021	To increase classification accuracy	TCN + LSTM	Python, Tensorflow
[13]	2022	To increase security and performance	DNN	Python, Numpy

Table 2. Best performance scores on the CSE-CIC-IDS2018 dataset.

Author	Accuracy	Precision	Recall	F1-Score
[8]	97.90%	98.00%	98.00%	98.00%
[9]	96.99%	97.46%	96.97%	97.04%
[10]	98.38%	97.79%	97.74%	97.76%
[11]	95.79%	95.38%	95.79%	95.11%
[12]	97.77%	97.94%	97.53%	97.73%
[13]	97.93%	99.97%	97.42%	98.68%

2.1. A Semi-Self-Taught Network Intrusion Detection System [8]

The authors used the denoising autoencoder (DAE) and the fuzzy c-means algorithm to build their IDS. This strategy was chosen to address missing values and duplicate data because the different class labels for duplicate or misclassified instances cause class noise. The ratio for training–validation–testing was 70%-15%-15%. The performance scores were 97.90% for accuracy and 98.00% for precision and recall. The limitations of this study include the lack of detail about the experiment and the use of only one classifier.

2.2. Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning [9]

The authors used a classical classification algorithm comprising machine learning (ML) and deep learning (DL), including random forest (RF), support vector machine (SVM), XGBoost, long short-term memory (LSTM), AlexNet, and Mini-VGGNet. The network was trained on the NSL-KDD and CSE-CIC-IDS2018 datasets. Data preprocessing involved the removal of the timestamp, destination address, source address, source port, etc., from the CSE-CIC-IDS2018. The training–testing ratio was 80%-20%. The highest accuracy was 96.99%, with a precision of 97.46% and 96.97% for recall. The shortcoming of this research is the absence of specific essential details about the experiments.

2.3. Deep Learning Methods in Network Intrusion Detection: A Survey and an Objective Comparison [10]

The authors used four models based on DNN, AE, deep belief network (DBN), and LSTM. These models were trained on KDD99, NSL-KDD, CIC-IDS2017, and CSE-CIC-IDS2018. The outcomes demonstrate the DNN model's efficacy on all four datasets. Concerning CSE-CIC-IDS2018, the highest accuracy for the DNN was 98.38%, corresponding to a precision of 97.79% and a recall of 97.74%. Some of the PCs involved in the experiments utilized GPU acceleration. The absence of performance scores for multi-class classification was a limitation of this study.

2.4. Attack Classification of an Intrusion Detection System Using Deep Learning and Hyperparameter Optimization [11]

The authors developed a hybrid deep-structured model using DAE for training and DNN for attack classification. They used an autoencoder as an input and a DNN as an output classifier. The raining–testing ratio was 10%-2.5%. Despite the contribution of this helpful study, the accuracy (95.79%) was the lowest among other related works, and the training–testing ratio was very small.

2.5. Network Anomaly Detection with Temporal Convolutional Network and U-Net Model [12]

The authors proposed three models to make the classification of network threats more accurate and to compare the datasets used in their study. The first model used a U-shaped network (U-Net), one of the convolutional neural networks (CNNs). The second model was based on a temporal convolutional network (TCN). The last model was based on a combination of TCN and LSTM. Furthermore, they trained the proposed models on the KDD99 and CSE-CIC-IDS2018 datasets. The ratio for training–validation–testing was 72%-18%-10%. The highest accuracy score was 97.77% based on the TCN-LSTM model, with a precision of 97.94% and 97.53% for recall. The lack of performance scores for binary classification was a limitation of this research.

2.6. Cyber Threat Intelligence Using PCA-DNN Model to Detect Abnormal Network Behavior [13]

The authors used two models based on DNN and DNN with principal component analysis (PCA). Furthermore, they used one-hot encoding and min–max scaling in the data preprocessing phase. They reduced the training time by half while maintaining the accuracy rate using DNN-PCA with only 12 features from the dataset. The training–testing ratio was 80%-20%. From the confusion matrix in the results, we can quickly achieve the highest accuracy of 97.93%, with a precision of 99.97% and a recall of 97.42%. A disadvantage was the absence of performance scores covering the multi-class classification.

3. Background

This section briefly presents a background of the IDS, DNN, PSO, and CSE-CIC-IDS2018 dataset, as shown in the following subsections.

3.1. Intrusion Detection System

An IDS is a piece of software, piece of hardware, or system that can detect harmful activity or protect the system from insider and outsider attacks on the installed device or network. Today, IDSs are an essential tool that network administrators use to protect their systems from cyber and intrusion attacks. The following Figure 1 shows the classification of cloud-based IDSs.

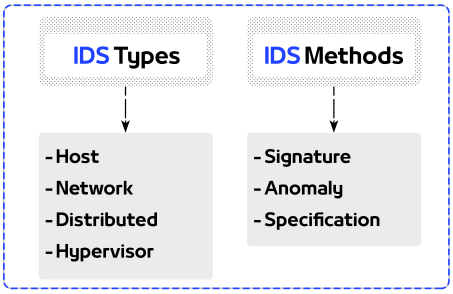


Figure 1. Classification of cloud-based IDSs.

3.1.1. Cloud-Based IDS Types

IDSs in the cloud can be classified into four main types, as follows:

1. Host IDS (HIDS): The HIDS operates by wiping and evaluating the records gathered from an individual computer system. The HIDS monitors system activities, such as system logs and CPU usage, to detect if the system has been attacked, and sends an alarm to the administrators [14].
2. Network IDS (NIDS): The NIDS observes and analyzes network flow and examines packet flow to detect suspicious activities, such as port scanning. NIDS efficiently detects network attacks, such as denial of service (DoS) [15].

3. Distributed IDS (DIDS): The DIDS contains various IDSs, for instance, HIDS and NIDS, that may communicate between themselves or with a centralized server for network monitoring. It may contain two main types of IDS methods. DIDS has the advantage of detecting known and unknown attacks [15].
4. Hypervisor IDS: The hypervisor IDS was created to run a virtual machine (VM), which observes communication between the VM and hypervisors. This type has the advantage of providing state information and event monitoring [16].

3.1.2. Cloud-Based IDS Methods

IDS in the cloud can be classified into two main methods, with a third possibility available, as follows:

1. Signature-based detection: This approach is indicated as misuse- or rule-based. It is based on documented signatures of previous attacks and aims to identify applicable intrusions by analyzing data in opposition to these predefined guidelines. Data compliance with the predefined guidelines could be identified as attacks and invoke an alarm. This system is not capable of detecting unknown attacks [17].
2. Anomaly-based detection: This system requires modern network users to be compared with a baseline of historically learned ordinary community behavior. The baseline is built using observations of the community and hosts for a prolonged period of profile creation. Any enormous perversion from this baseline is identified as an anomaly and invokes an alert. This system is capable of detecting unknown attacks [17].
3. Specification-based detection: This system is a hybrid of signature and anomaly, using the signature approach to detect known attacks and the anomaly approach to detect unknown attacks [18].

3.2. Deep Neural Network

ANNs are computer programs that are biologically inspired and are intended to imitate how the human brain processes information and obtains knowledge by identifying the patterns and correlations included within the data through experience [19]. ANNs comprise several individual units called neurons, which are coupled with weights to create the neural structure of the network, arranged in layers [20].

Nowadays, the term DNN is used to refer to a neural network with multiple hidden layers [21], and DNNs are categorized as a subset of ML [22,23]. Figure 2 shows the relationship between artificial intelligence (AI), ML, and DL.

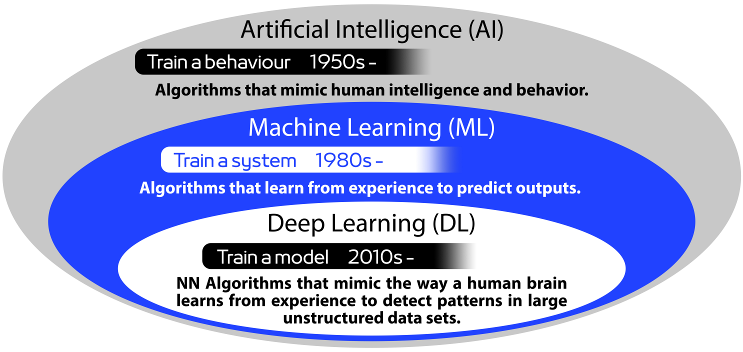

Artificial Intelligence (AI)
Train a behaviour 1950s -
Algorithms that mimic human intelligence and behavior.
Machine Learning (ML)
Train a system 1980s -
Algorithms that learn from experience to predict outputs.
Deep Learning (DL)
Train a model 2010s -
NN Algorithms that mimic the way a human brain learns from experience to detect patterns in large unstructured data sets.

Figure 2. Relationship between AI, ML, and DL.

In a DNN, when the weights of hidden layers are fully connected, this is called MLP [24]. In MLP, each neuron of the network has an activation function (such as softsign, tanh, relu, etc.). MLPs are categorized under feedforward algorithms in which the data flow from the input, through the hidden layers, to the output in one direction [25].

If the algorithm just calculates the weighted sum in each neuron and moves to the next layer, it cannot learn the weights used to minimize the cost function. Unless the algorithm iterates, there is no effective learning. To do this, we need to use the BP as a learning algorithm, which allows the MLP to adjust the network's weights iteratively by using gradient descent to make the cost function as low as possible [26,27].

To summarize, DNN refers to an ANN with multiple hidden layers [28], and feed-forward refers to the ANN's data flow in one direction from the input to the output [29]. MLP refers to an ANN with fully connected neurons and the use of a kind of activation function [30]. Using gradient descent, BP allows the MLP to adjust the weights to iteratively minimize the cost function [31]. Figure 3 describes the architecture of MLP.

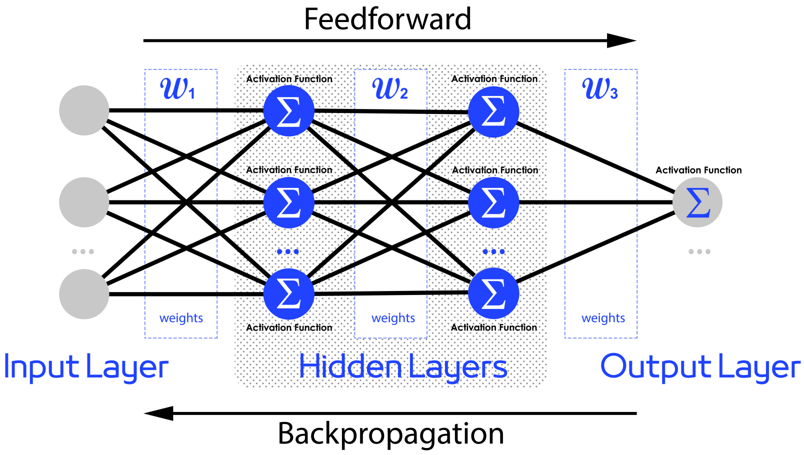


Figure 3. MLP: feedforward DNN with BP and two hidden layers.

3.3. Particle Swarm Optimization

In 1995, Kennedy and Eberhart proposed the PSO algorithm [32], the goal of which was to solve optimization problems. It is a metaheuristic algorithm categorized under swarm intelligence algorithms [33]. The concept of PSO is to imitate the behavior of birds to acquire food [34]. Each bird modifies its position in the search space to achieve the best solution. Any time a bird finds a food source, it will head toward it [35]. Over the generations (iterations), every particle searches for its own optimal solution and the optimal solution in the swarm. Then, it updates the particle's position (p) and the rate of change associated with the position (velocity). These updates will affect the personal best (pbest), and global best (gbest) values [36].

The following Equation (1) shows how the velocity is dynamically adjusted for each particle:

$$v_k^{t+1} = v_k^t + c_1 r_1 \left(pbest_k^t - p_k^t \right) + c_2 r_2 (gbest^t - p_k^t) \tag{1}$$

The velocity in the $(t+1)$th iteration is affected by the three variables listed in Table 3 below [37].

Table 3. Velocity components.

Notation	Component
v_k^t	physical component
$c_1 r_1 \left(pbest_k^t - p_k^t \right)$	cognitive component
$c_2 r_2 (gbest^t - p_k^t)$	social component

The position equation, Equation (2), can be written as follows:

$$p_k^{t+1} = p_k^t + v_k^{t+1} \tag{2}$$

Table 4 shows the main parameters used in Equations (1) and (2).

Table 4. Parameters of PSO.

Parameter	Description
p_k	the position of a particle
v_k	the velocity of a particle
$pbest_k$	the best solution for the particle
$gbest_k$	the best solution for the swarm
c_1	cognitive factor
c_2	social factor
r_1, r_2	random numbers between 0 and 1
t	the iteration number

Figure 4 shows how the moving particle looks in the search space, and Figure 5 shows the steps of the PSO algorithm.

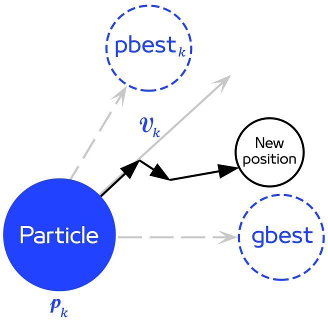


Figure 4. Moving particle in PSO.

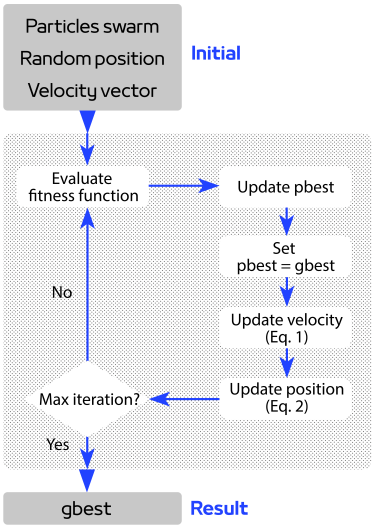


Figure 5. Steps of the PSO algorithm.

3.4. CSE-CIC-IDS2018 Dataset

The CSE-CIC-IDS2018 dataset is a collaboration project proposed in 2018 by CSE [38] and CIC [39]. The dataset is available to download on Amazon Web Services (AWS) [40,41].

The dataset contains ten CSV files representing ten days from the captured network flow, with more than 16.2 million samples. Furthermore, more than 80 features were extracted by the CICFlowMeter tool.

This dataset includes six primary intrusion attack types: distributed DoS (DDoS), DoS, brute force, bot, infiltration, and web attacks, as shown in Tables 5 and 6 and Figure 6.

Table 5. The network traffic of CSE-CIC-IDS2018.

Day	File Name	Features	Type of Attack	Count	Total
1	Wednesday-14-02-2018	80	Benign FTP-BruteForce SSH-Bruteforce	667,626 193,360 187,589	1,048,575
2	Thursday-15-02-2018	80	Benign DoS attacks-GoldenEye DoS attacks-Slowloris	996,077 41,508 10,990	1,048,575
3	Friday-16-02-2018	80	Benign DoS attacks-Hulk DoS attacks-SlowHTTPTest	446,772 461,912 139,890	1,048,574
4	Thursday-20-02-2018	84	Benign DDoS attacks-LOIC-HTTP	7,372,557 576,191	7,948,748
5	Wednesday-21-02-2018	80	Benign DDoS attack-HOIC DDoS attack-LOIC-UDP	360,833 686,012 1730	1,048,575
6	Thursday-22-02-2018	80	Benign Brute Force -Web Brute Force -XSS SQL Injection	1,048,213 249 79 34	1,048,575
7	Friday-23-02-2018	80	Benign Brute Force -Web Brute Force -XSS SQL Injection	1,048,009 362 151 53	1,048,575
8	Wednesday-28-02-2018	80	Benign Infilteration	544,200 68,871	613,071
9	Thursday-01-03-2018	80	Benign Infilteration	238,037 93,063	331,100
10	Friday-02-03-2018	80	Benign Botnet	762,384 286,191	1,048,575

Total of rows (Samples) = 16,232,943; Total of columns (Features) = 84.

Table 6. Primary attack classes of CSE-CIC-IDS2018.

Class	Benign	DDoS	DoS	Brute Force	Bot	Infilteration	Web
Type	Benign	HOIC LOIC-UDP LOIC-HTTP	Hulk GoldenEye Slowloris SlowHTTPTest	FTP SSH	Botnet	Infiltration	Web XSS SQL Injection
Count	13,484,708	1,263,933	654,300	380,949	286,191	161,934	928

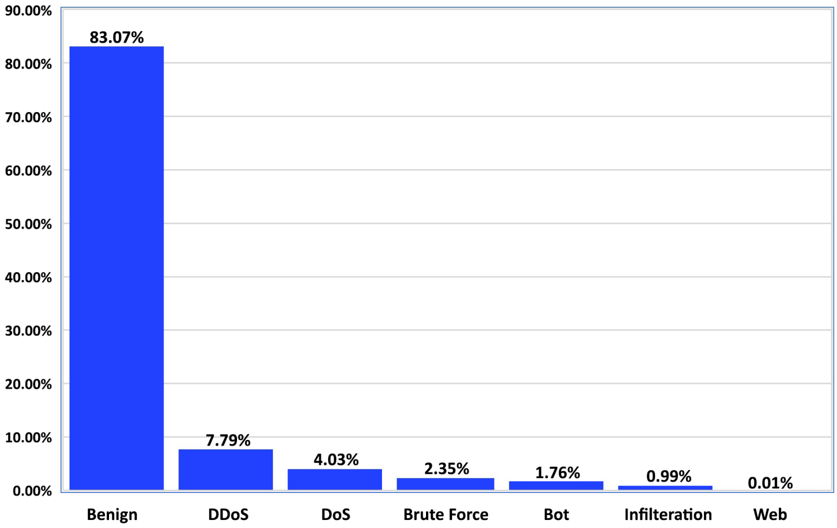


Figure 6. Distribution of attack classes in CSE-CIC-IDS2018.

4. Methodology

According to the study's goals, we aimed to build robust and efficient IDSs in the cloud environment. We proposed two models, the first using MLP-BP and the other using MLP-PSO, and used the CSE-CIC-IDS-2018 dataset to train them. The following subsections describe the research design and experimental setup.

4.1. Research Framework and Proposed Methods

This project comprises three phases, and the output from each phase will come as input to the next, as shown in the following Figure 7.

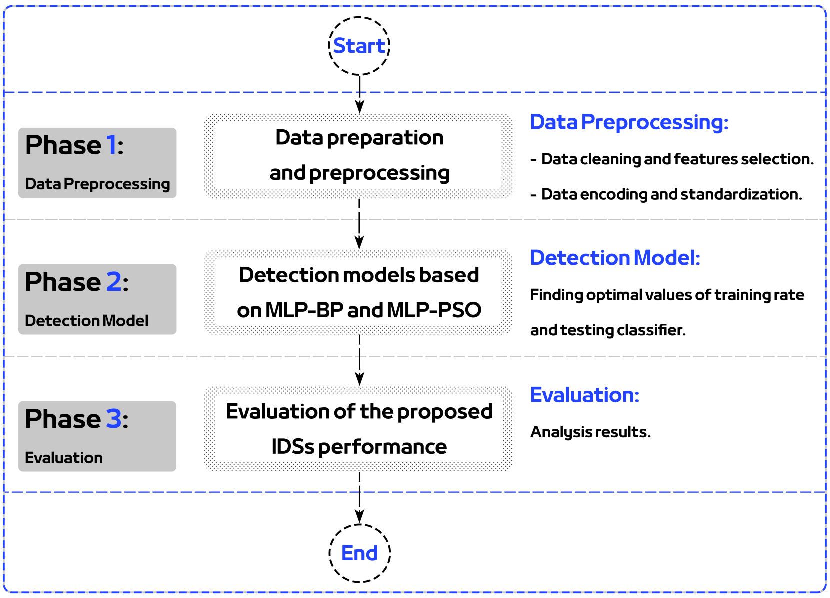


Figure 7. The framework of research.

- Phase 1: Data preprocessing and cleaning are performed in this phase, including missing, duplicate, infinity, and NaN values. Furthermore, it includes processes such as dropping unnecessary features, feature importance, feature selection, data encoding, and data standardization.
- Phase 2: In this phase, we will use four scenarios for the training based on DNNs. We will build two models, the first using MLP-BP and the other using MLP-PSO for both binary and multi-class classification.

- Phase 3: We will use the confusion matrix and equations of performance metrics to evaluate the outcomes of our models.

The flowchart of the proposed methodology is illustrated in Figure 8, and Figure 9 describes the frameworks of our models.

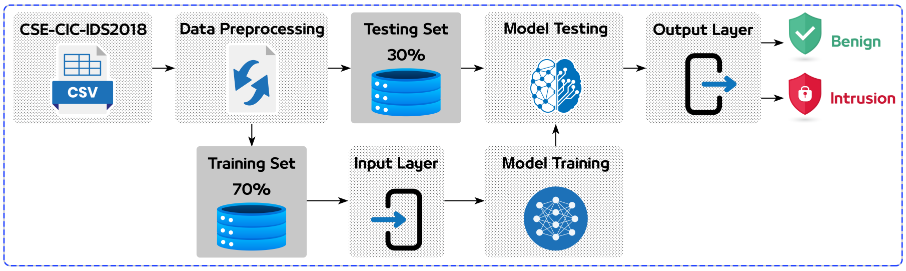


Figure 8. General methodology workflow.

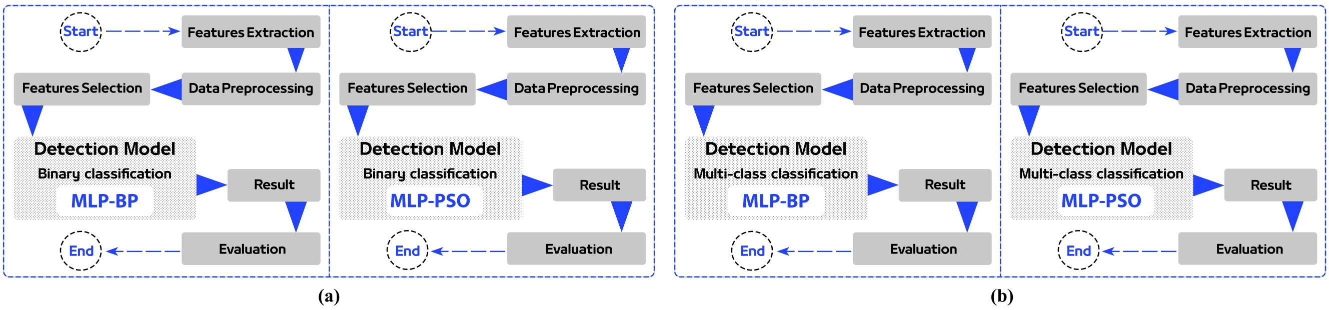


Figure 9. The proposed models' frameworks: (**a**) binary classification; (**b**) multi-class classification.

4.2. Data Preparation and Preprocessing

Firstly, we combined the CSV files into one using JupyterLab and Python with the Pandas and Numpy libraries. Furthermore, we cleaned the dataset by removing positive and negative infinity and NaN values. After that, we dropped unnecessary features ("Timestamp", "Flow ID", "Src IP", "Src Port", and "Dst IP"). In the next step, from the Scikit-learn library, we used the random forest classifier to compute the feature importance, as illustrated in Figure 10.

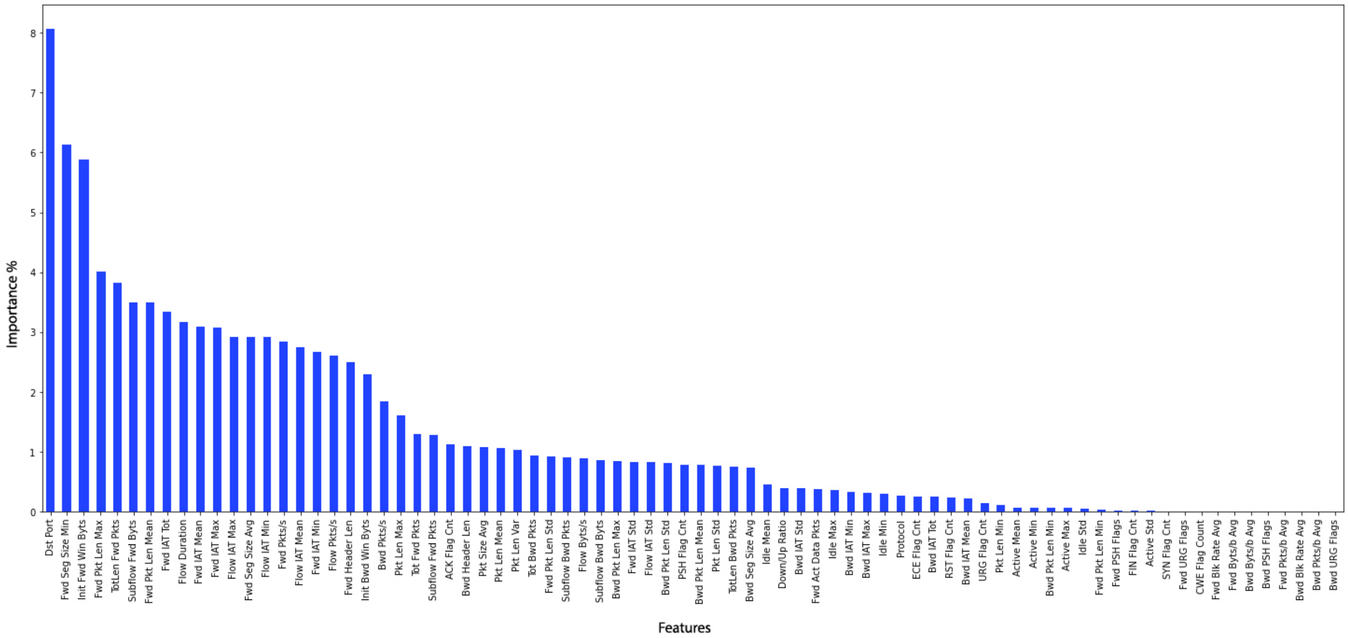


Figure 10. Feature importance ranking of CSE-CIC-IDS2018.

In contrast, from the Scikit-learn library, we used the SelectFromModel class for feature selection. The outcome of this process was that 24 features were selected, as displayed in Table 7.

Table 7. List of selected features.

Rank	Feature	Rank	Feature	Rank	Feature
1	Dst Port	9	Flow Duration	17	Fwd IAT Min
2	Fwd Seg Size Min	10	Fwd IAT Mean	18	Flow Pkts/s
3	Init Fwd Win Byts	11	Fwd IAT Max	19	Fwd Header Len
4	Fwd Pkt Len Max	12	Flow IAT Max	20	Init Bwd Win Byts
5	TotLen Fwd Pkts	13	Fwd Seg Size Avg	21	Bwd Pkts/s
6	Subflow Fwd Byts	14	Flow IAT Min	22	Pkt Len Max
7	Fwd Pkt Len Mean	15	Fwd Pkts/s	23	Tot Fwd Pkts
8	Fwd IAT Tot	16	Flow IAT Mean	24	Subflow Fwd Pkts

Afterward, we made two copies of the dataset. The first copy was for binary classification, and the second was for multi-class classification, containing the following classes: Benign, DDoS, DoS, BruteForce, Bot, Infilteration, and Web.

In the final step, we applied OneHotEncoder to the label column (Y) to convert the categorical data into numerical data, differentiating each class from the others. This process is essential because the neural network requires the data to be numeric. Finally, we applied StandardScaler to the other columns (X); the data had a mean value of 0 and a standard deviation of 1. In other words, after applying StandardScaler, the original distribution of the data remains the same, with the range restricted to [0, 1]. This process is helpful in classification.

4.3. Experimental Setup

Our experiments were applied on AWS, which provides many services, such as the elastic compute cloud (EC2). From EC2, we used an instance size of p3.8xlarge to train our models with GPUs. Accordingly, we used TensorFlow to distribute training across multiple GPUs using the tf.distribute.Strategy API [42]. Furthermore, we used the Pyswarms toolkit package [43] in our proposed MLP-PSO model.

In this subsection, we will summarize our DNN structure and all of the main tools used in this study, as given in the following tables and figures. Figure 11 describes our DNN architecture.

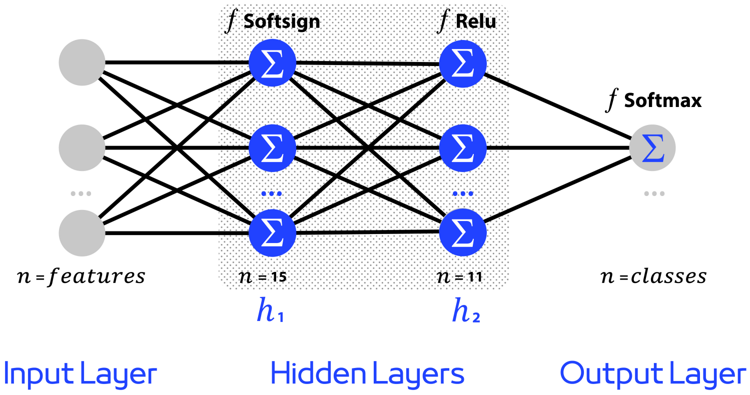


Figure 11. The architecture of our DNN.

Our DNN parameters are shown in Table 8. Furthermore, Table 9 lists the hardware resources and software environments used in this study. The most important libraries, classes, modules, functions, and algorithms used in our experiments are shown in Table 10.

Table 8. Our DNN parameters.

Parameter	Value
Hidden layers (h)	2
n	Number of neurons
$n\ features$	24
$n\ classes$	2 neurons in binary classification 7 neurons in multi-class classification
Loss function	"BinaryCrossentropy" for binary classification "CategoricalCrossentropy" for multi-class classification
Optimization algorithm	Adam
Learning rate	Using ExponentialDecay
Batch size	2048
Epochs	100
Training-testing ratio	70% (11,363,060 samples)–30% (4,869,883 samples)
Cross-validation	5 folds
PSO iterations	50
PSO swarm	25
PSO c_1	1.5
PSO c_2	1.5
PSO w	0.9

Table 9. List of hardware resources and software environments.

Hardware/Software	Detail
Computing platform	Platform: Amazon Web Services (AWS) Instance size: p3.8xlarge OS: Ubuntu 20.04.4 LTS x86_64 CPU: Intel Xeon E5-2686 v4 3.00GHz GPUs: 4x (NVIDIA Tesla V100 SXM2 16GB) vCPUs: 32 RAM: 244 (GiB)
Programming language	Python
Web-based IDE	JupyterLab
ML framework	Scikit-learn
DL framework	TensorFlow-GPU
DL API	Keras

Table 10. List of programming tools.

List 1	List 2	List 3
math	keras.layers	PyCM
keras	keras.models	KFold
numpy	keras.callbacks	EarlyStopping
pandas	sklearn.metrics	GlobalBestPSO
sklearn	keras.optimizers	StandardScaler
seaborn	sklearn.ensemble	train_test_split
datetime	sklearn.preprocessing	OneHotEncoder
matplotlib	sklearn.neural_network	classification_report
pyswarms	sklearn.model_selection	LearningRateScheduler
Tensorflow	sklearn.feature_selection	RandomForestClassifier

5. Results

We will use the following equations in Table 11 to evaluate the performance metrics for binary classification, and those in Table 12 to evaluate the performance metrics for multi-class classification [44–46].

Table 11. Equations of the performance metrics for binary classification.

Measure	Formula
Accuracy	Acc = (TP + TN)/Total
Misclassification Rate (Error Rate)	ERR = 1-Acc
Precision (Positive Predictive Value)	PPV = TP/(TP + FP)
Recall (Sensitivity or True Positive Rate)	TPR = TP/(TP + FN)
Specificity (True Negative Rate)	TNR = TN/(TN + FP)
F1-score	F1 = 2(PPV * TPR)/(PPV + TPR)
False Positive Rate (Type I Error)	FPR = FP/(TN + FP) or FPR = 1-Specificity
False Negative Rate (Type II Error)	FNR = FN/(TP + FN) or FNR = 1-Sensitivity

Table 12. Equations of the performance metrics for multi-class classification.

Measure	Formula
Accuracy	$\text{Acc} = \frac{\sum TP}{Total}$
Precision	$\text{PPV} = \sum \frac{TP}{(TP + FP)}$
Recall	$\text{TPR} = \sum \frac{TP}{(TP + FN)}$

5.1. MLP-BP (Binary Classification)

Table 13 describes the average accuracy of the proposed models for binary classification using five-fold cross-validation.

Table 13. The average accuracy of proposed models for binary classification.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average Accuracy
MLP-BP	98.95%	98.97%	98.99%	98.98%	98.97%	98.97%
MLP-PSO	96.23%	96.24%	96.25%	96.26%	96.25%	96.25%

Figures 12 and 13 show the training and testing accuracy and the confusion matrix of MLP-BP for binary classification. Table 14 provides the performance metric scores.

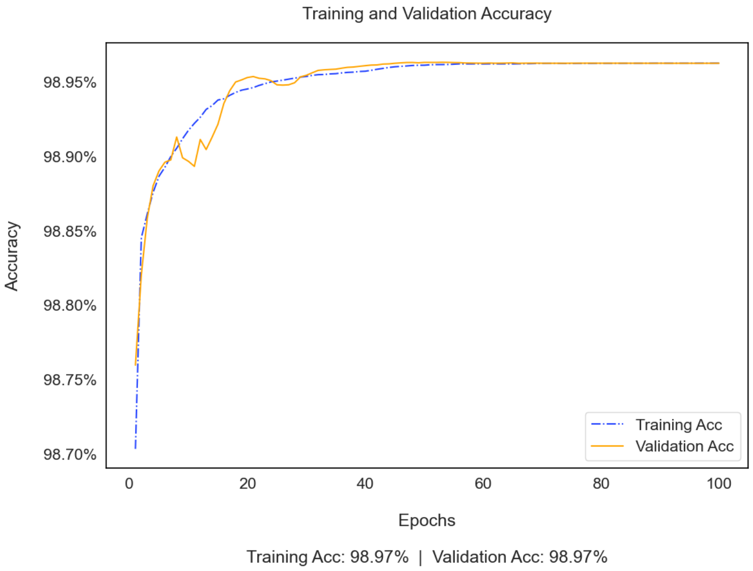


Figure 12. Accuracy of MLP-BP for binary classification.

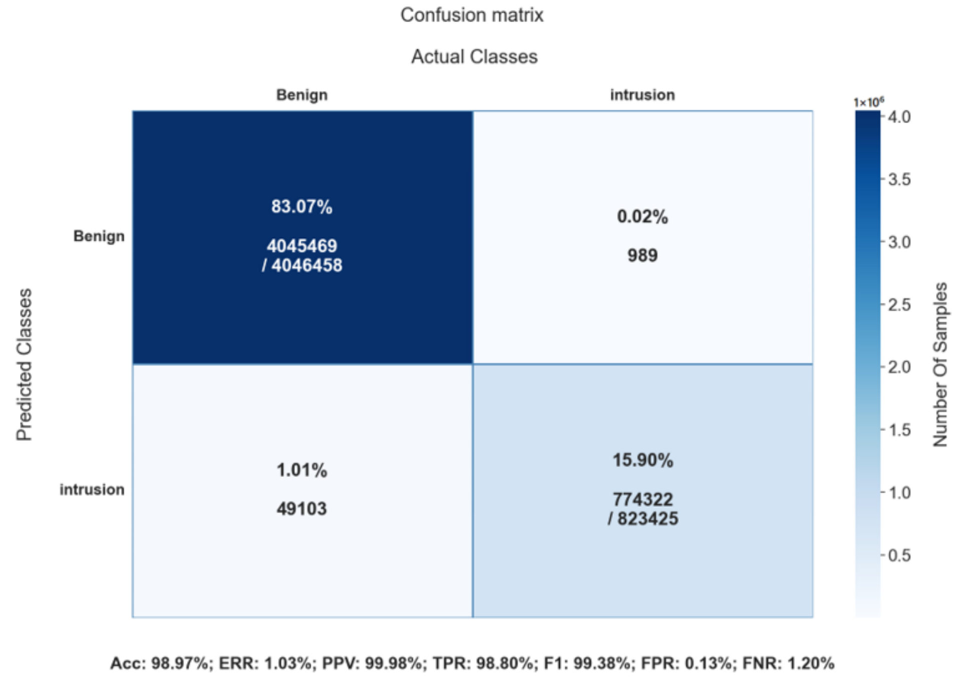


Figure 13. Confusion matrix of MLP-BP for binary classification.

Table 14. Performance scores for each class of MLP-BP for binary classification.

Class	Precision	Recall	F1-Score	Samples
Benign	0.99	1.00	0.99	4,046,458
Intrusion	1.00	0.94	0.97	823,425

5.2. MLP-PSO (Binary Classification)

Figures 14 and 15 show the training and testing accuracy and the confusion matrix of MLP-PSO for binary classification. Table 15 provides the performance metric scores.

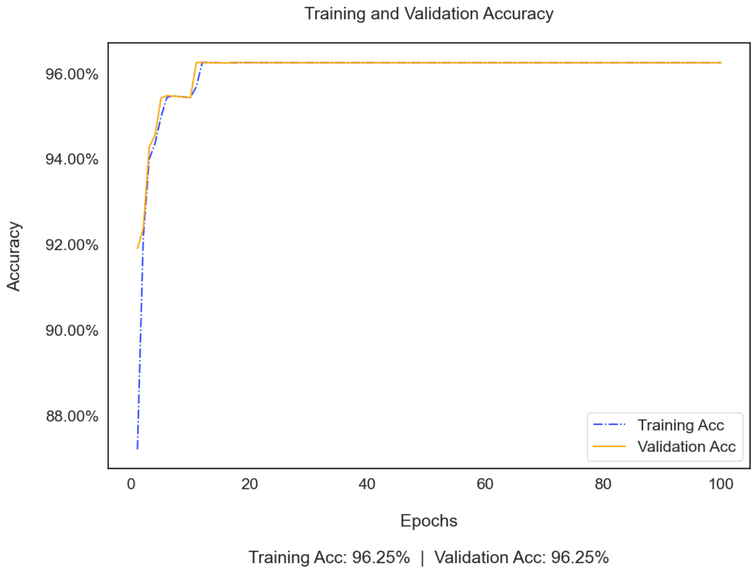


Figure 14. Accuracy of MLP-PSO for binary classification.

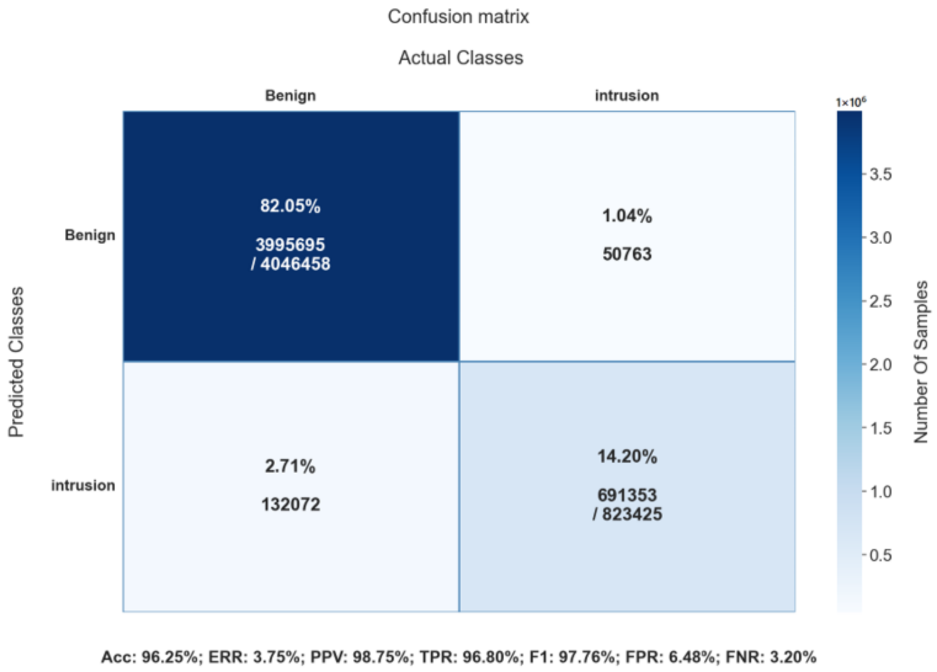


Figure 15. Confusion matrix of MLP-PSO for binary classification.

Table 15. Performance scores for each class of MLP-PSO for binary classification.

Class	Precision	Recall	F1-Score	Samples
Benign	0.97	0.99	0.98	4,046,458
Intrusion	0.93	0.84	0.88	823,425

5.3. MLP-BP (Multi-Class Classification)

In addition to the binary classification, we show the multi-class classification results with the MLP-BP and MLP-PSO models. Table 16 shows the average accuracy of the proposed models for multi-class classification using five-fold cross-validation.

Table 16. The average accuracy of proposed models for multi-class classification.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average Accuracy
MLP-BP	98.40%	98.39%	98.41%	98.42%	98.41%	98.41%
MLP-PSO	95.31%	95.32%	95.32%	95.31%	95.32%	95.32%

Table 17 describes the scores of performance metrics for each class of MLP-BP for multi-class classification.

Table 17. Performance scores for each class of MLP-BP for multi-class classification.

Class	Precision	Recall	F1-Score	Samples
Benign	0.99	1.00	0.99	4,046,458
DDoS	1.00	1.00	1.00	378,558
DoS	0.96	0.90	0.93	196,350
BruteForce	0.84	0.94	0.89	113,902
Bot	1.00	1.00	1.00	85,680
Infilteration	0.44	0.01	0.01	48,649
Web	0.95	0.24	0.38	286

Figures 16 and 17 show the training and testing accuracy and the confusion matrix of MLP-BP for multi-class classification.

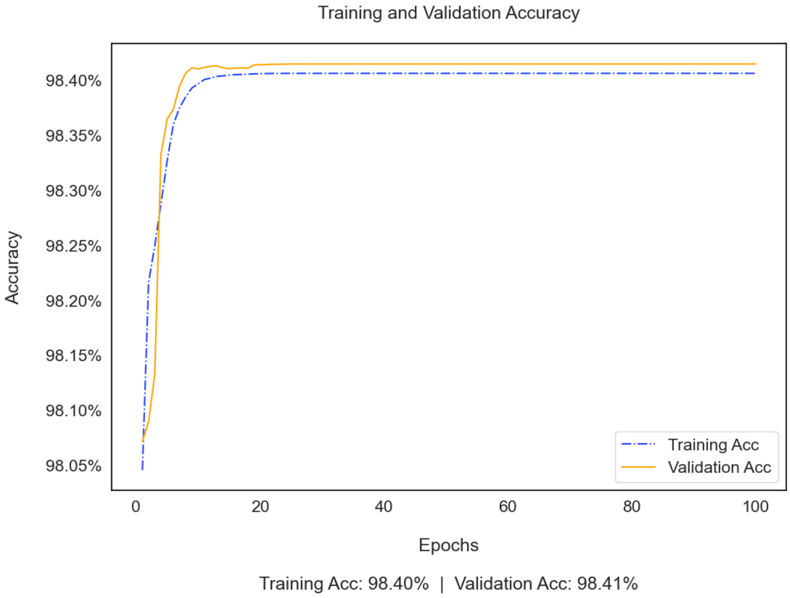


Figure 16. Accuracy of MLP-BP for multi-class classification.

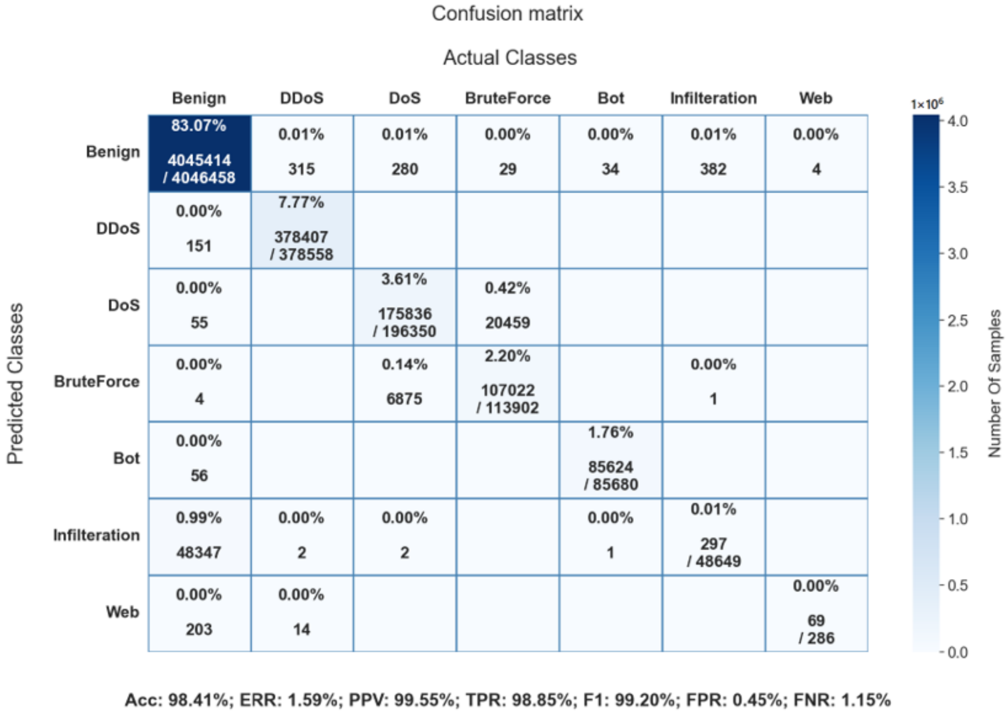


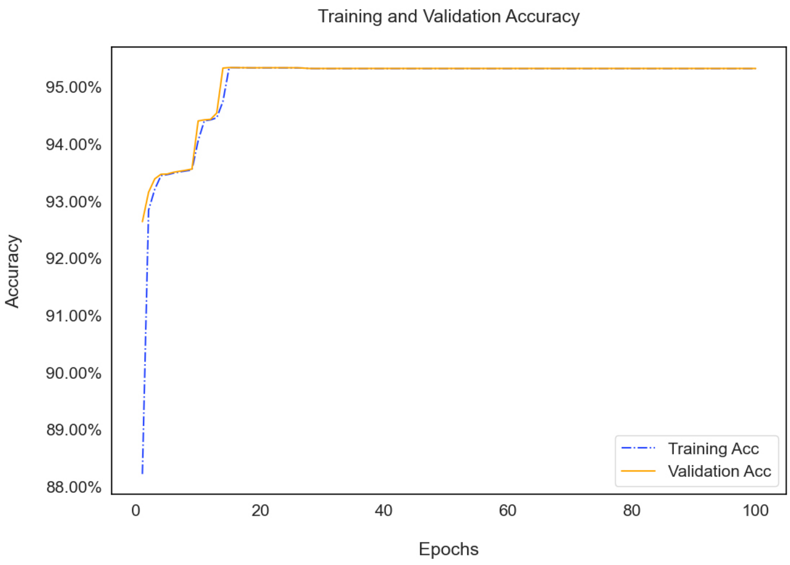
Figure 17. Confusion matrix of MLP-BP for multi-class classification.

5.4. MLP-PSO (Multi-Class Classification)

Table 18 describes the performance metric scores for each class of MLP-PSO for multi-class classification. Figures 18 and 19 show the training and testing accuracy and the confusion matrix of MLP-PSO for multi-class classification.

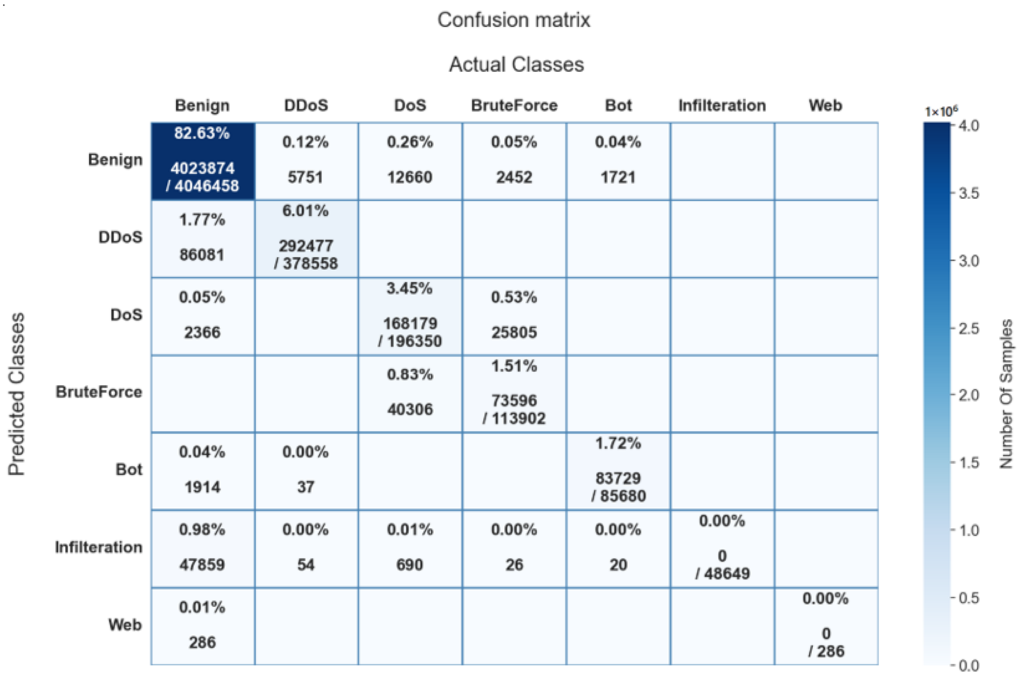
Table 18. Performance scores for each class of MLP-PSO for multi-class classification.

Class	Precision	Recall	F1-Score	Samples
Benign	0.97	0.99	0.98	4,046,458
DDoS	0.98	0.77	0.86	378,558
DoS	0.76	0.86	0.80	196,350
BruteForce	0.72	0.65	0.68	113,902
Bot	0.98	0.98	0.98	85,680
Infilteration	0.00	0.00	0.00	48,649
Web	0.00	0.00	0.00	286



Training Acc: 95.32% | Validation Acc: 95.32%

Figure 18. Accuracy of MLP-PSO for multi-class classification.



Acc: 95.32%; ERR: 4.68%; PPV: 98.97%; TPR: 96.27%; F1: 97.60%; FPR: 1.03%; FNR: 3.73%

Figure 19. Confusion matrix of MLP-PSO for multi-class classification.

6. Discussion

The previous results show that the MLP-PSO model performs worse for binary and multi-class classification, with accuracy scores of 96.25% and 95.32%. Conversely, the MLP-BP obtains better accuracy scores (98.97% and 98.41%) for binary and multi-class classification. Furthermore, the MLP-BP obtains better results for FPR, with a value of 0.13% for binary classification and 0.45% for multi-class classification. Furthermore, FNR scores 1.20% and 1.15% for binary and multi-class classification, respectively. Although both types of errors are detrimental to IDSs, FNR (type II error) is more damaging and dangerous since it incorrectly classifies the intrusion attacks as benign. Table 19 summarizes the overall performance scores of the proposed models.

We point out that these results may differ according to the dataset or the strategy. Through our experiments in this study, we find that TensorFlow and Keras have sufficient tools to achieve perfect results without needing an external optimization algorithm. In other situations, it may be helpful to use an optimization algorithm to achieve the best results.

Furthermore, the PSO does not achieve better scores and requires a great deal of training time and hardware resources. Therefore, we suggest building a DNN with TensorFlow and Keras and evaluating the performance before using an external optimization algorithm. After that, according to the outcomes, it can be determined if an optimization algorithm is required.

Additionally, we advise adjusting the learning rate and trying several times to achieve excellent scores and converge toward the minimum in a less noisy manner. TensorFlow and Keras provide several methods to achieve this, such as LearningRateScheduler and ExponentialDecay.

The best performance scores of the proposed MLP-BP and MLP-PSO models compared with those in related works are given in Table 20.

Table 19. Overall performance scores of the proposed models.

Model	Accuracy	ERR	Precision	Recall	F1-Score	FPR	FNR
MLP-BP (Binary Classification)	98.97%	1.03%	99.98%	98.80%	99.38%	0.13%	1.20%
MLP-PSO (Binary Classification)	96.25%	3.75%	98.75%	96.80%	97.76%	6.48%	3.20%
MLP-BP (Multi-class Classification)	98.41%	1.59%	99.55%	98.85%	99.20%	0.45%	1.15%
MLP-PSO (Multi-class Classification)	95.32%	4.68%	98.97%	96.27%	97.60%	1.03%	3.73%

Table 20. The best performance scores of the proposed models compared with related works.

Model	Accuracy	Precision	Recall	F1-Score
[8]	97.90%	98.00%	98.00%	98.00%
[9]	96.99%	97.46%	96.97%	97.04%
[10]	98.38%	97.79%	97.74%	97.76%
[11]	95.79%	95.38%	95.79%	95.11%
[12]	97.77%	97.94%	97.53%	97.73%
[13]	97.93%	99.97%	97.42%	98.68%
Proposed MLP-PSO	96.25%	98.75%	96.80%	97.76%
Proposed MLP-BP	98.97%	99.98%	98.80%	99.38%

7. Conclusions and Outlooks

In this study, we conducted four experiments based on the MLP-BP and MLP-PSO models for binary and multi-class classification on the CSE-CIC-IDS-2018 dataset. We presented a list of related works and discussed them. Furthermore, we provided an analysis of the CSE-CIC-IDS-2018 dataset and details of our DNN architecture, data preparation, data preprocessing, feature importance, feature selection, and proposed models.

Additionally, we described the details of the hardware resources and software environments used in this study, in addition to the most important libraries, classes, modules, functions, and algorithms used in our experiments.

Finally, we provided and discussed the results. The best accuracy for binary classification was 98.97% and that for multi-class classification was 98.41% based on MLP-BP. Furthermore, we tried to provide some essential tips that we think are useful to researchers in this field. Furthermore, our proposed models showed an improvement in performance metrics scores, which we compared with those in related works.

In the future, we plan to run our IDSs in a cloud computing environment and evaluate the detection efficiency in real time. Furthermore, we propose using another approach or a different metaheuristic optimization algorithm to improve the results.

Author Contributions: Methodology, S.A.; Software, S.A.; Validation, S.E.K.; Writing—original draft, S.A.; Writing—review & editing, S.E.K.; Supervision, S.E.K. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge Qassim University, represented by the Deanship of "Scientific Research, on the financial support for this research under the number (COC-2022-1-2-J-29853) during the academic year 1444 AH / 2022 AD".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

CC	Cloud Computing
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
BP	Backpropagation
RF	Random Forest
VM	Virtual Machine
AE	Autoencoder
DAE	Denoising Autoencoder
IDS	Intrusion Detection System
ANN	Artificial Neural Network
DNN	Deep Neural Network
DBN	Deep Belief Network
CNN	Convolutional Neural Network
TCN	Temporal Convolutional Network
U-Net	U-shaped Network
MLP	Multi-Layer Perceptron
PSO	Particle Swarm Optimization
SVM	Support Vector Machine
PCA	Principal Component Analysis
CSE	Communications Security Establishment
CIC	Canadian Institute for Cybersecurity
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
IDE	Interactive Development Environment
Acc	Accuracy
ERR	Error Rate
PPV	Positive Predictive Value
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
FNR	False Negative Rate
DoS	Denial of Service

DDoS	Distributed Denial of Service
LSTM	Long Short-Term Memory
HIDS	Host Intrusion Detection System
NIDS	Network Intrusion Detection System
DIDS	Distributed Intrusion Detection System

References

1. Saljoughi, A.S.; Mehrvarz, M.; Mirvaziri, H. Attacks and intrusion detection in cloud computing using neural networks and particle swarm optimization algorithms. *Emerg. Sci. J.* **2017**, *1*, 179–191. [CrossRef]
2. Riaz, A.; Ahmad, H.F.; Kiani, A.; Qadir, J.; Rasool, R.; Younis, U. Intrusion detection systems in cloud computing: A contemporary review of techniques and solutions. *J. Inf. Sci. Eng.* **2017**, *33*, 611–634. [CrossRef]
3. Çavuşoğlu, Ü. A new hybrid approach for intrusion detection using machine learning methods. *Appl. Intell.* **2019**, *49*, 2735–2761. [CrossRef]
4. Chong, H.Y.; Yap, H.J.; Tan, S.C.; Yap, K.S.; Wong, S.Y. Advances of metaheuristic algorithms in training neural networks for industrial applications. *Soft Comput.* **2021**, *25*, 11209–11233. [CrossRef]
5. Shi, Y. Particle swarm optimization. *IEEE Connect.* **2004**, *2*, 8–13.
6. Rehman, M.Z.; Nawi, N.M. The effect of adaptive momentum in improving the accuracy of gradient descent back propagation algorithm on classification problems. In Proceedings of the International Conference on Software Engineering and Computer Systems, Kuantan, Malaysia, 27–29 June 2011; pp. 380–390. [CrossRef]
7. Rezaei, F.; Safavi, H.R. GuASPSO: A new approach to hold a better exploration–exploitation balance in PSO algorithm. *Soft Comput.* **2020**, *24*, 4855–4875. [CrossRef]
8. Zhao, F.; Zhang, H.; Peng, J.; Zhuang, X.; Na, S.G. A semi-self-taught network intrusion detection system. *Neural Comput. Appl.* **2020**, *32*, 17169–17179. [CrossRef]
9. Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access* **2020**, *9*, 7550–7563. [CrossRef]
10. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. [CrossRef]
11. Kunang, Y.N.; Nurmaini, S.; Stiawan, D.; Suprapto, B.Y. Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *J. Inf. Secur. Appl.* **2021**, *58*, 102804. [CrossRef]
12. Mezina, A.; Burget, R.; Travieso-González, C.M. Network anomaly detection with temporal convolutional network and U-Net model. *IEEE Access* **2021**, *9*, 143608–143622. [CrossRef]
13. Al-Fawa'reh, M.; Al-Fayoumi, M.; Nashwan, S.; Fraihat, S. Cyber threat intelligence using PCA-DNN model to detect abnormal network behavior. *Egypt. Inform. J.* **2022**, *23*, 173–185. [CrossRef]
14. Elmaaradi, A.; Lyhyaoui, A.; Chairi, I. New security architecture using hybrid IDS for virtual private clouds. In Proceedings of the 2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS), Marrakech, Morocco, 28–30 October 2019; pp. 1–5. . [CrossRef]
15. Mehmood, Y.; Shibli, M.A.; Habiba, U.; Masood, R. Intrusion detection system in cloud computing: Challenges and opportunities. In Proceedings of the 2013 2nd National Conference on Information Assurance (NCIA), Rawalpindi, Pakistan, 11–12 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 59–66. [CrossRef]
16. Mahajan, V.; Peddoju, S.K. Deployment of intrusion detection system in cloud: A performance-based study. In Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, NSW, Australia, 1–4 August 2017; pp. 1103–1108. . [CrossRef]
17. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A survey of intrusion detection techniques in cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57. [CrossRef]
18. Elrawy, M.F.; Awad, A.I.; Hamed, H.F. Intrusion detection systems for IoT-based smart environments: A survey. *J. Cloud Comput.* **2018**, *7*, 1–20. [CrossRef]
19. Malekian, A.; Chitsaz, N. Concepts, procedures, and applications of artificial neural network models in streamflow forecasting. In *Advances in Streamflow Forecasting*; Elsevier: Amsterdam, The Netherlands, 2021; pp. 115–147. . [CrossRef]
20. Agatonovic-Kustrin, S.; Beresford, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J. Pharm. Biomed. Anal.* **2000**, *22*, 717–727. . [CrossRef]
21. Kim, D.E.; Gofman, M. Comparison of shallow and deep neural networks for network intrusion detection. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 204–208. [CrossRef]
22. Gupta, R.; Srivastava, D.; Sahu, M.; Tiwari, S.; Ambasta, R.K.; Kumar, P. Artificial intelligence to deep learning: Machine intelligence approach for drug discovery. *Mol. Divers.* **2021**, *25*, 1315–1360. [CrossRef] [PubMed]
23. Ceron, R. AI, Machine Learning and Deep Learning: What's the Difference. IBM IT Infrastructure Blog. 2019. Available online: https://www.ibm.com/blogs/systems/ai-machine (accessed on 2 Feburary 2023).
24. Deng, L.; Yu, D. Deep learning: Methods and applications. *Found. Trends® Signal Process.* **2014**, *7*, 197–387. [CrossRef]
25. Zhang, Z.; Zhang, K.; Khelifi, A. *Multivariate TIME Series Analysis in Climate and Environmental Research*; Springer: Berlin/Heidelberg, Germany, 2018. [CrossRef]

26. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
27. Rojas, R. The backpropagation algorithm. In *Neural Networks*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 149–182. [CrossRef]
28. Li, L.; Zhao, Y.; Jiang, D.; Zhang, Y.; Wang, F.; Gonzalez, I.; Valentin, E.; Sahli, H. Hybrid deep neural network–Hidden markov model (dnn-hmm) based speech emotion recognition. In Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, Geneva, Switzerland, 2–5 September 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 312–317. [CrossRef]
29. Poznyak, T.; Oria, J.I.C.; Poznyak, A. *Ozonation and Biodegradation in Environmental Engineering: Dynamic Neural Network Approach*; Elsevier: Amsterdam, The Netherlands, 2019. [CrossRef]
30. Camacho Olmedo, M.; Paegelow, M.; Mas, J.F.; Escobar, F. Geomatic approaches for modeling land change scenarios. An introduction. In *Geomatic Approaches for Modeling Land Change Scenarios*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 1–8. [CrossRef]
31. Aggarwal, C.C. Training deep neural networks. In *Neural Networks and Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 105–167. [CrossRef]
32. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95–International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
33. Mavrovouniotis, M.; Li, C.; Yang, S. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm Evol. Comput.* **2017**, *33*, 1–17. [CrossRef]
34. Carvalho, M.; Ludermir, T.B. An analysis of PSO hybrid algorithms for feed-forward neural networks training. In Proceedings of the 2006 Ninth Brazilian Symposium on Neural Networks (SBRN'06), Ribeirao Preto, Brazil, 23–27 October 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 6–11. [CrossRef]
35. Shelokar, P.; Siarry, P.; Jayaraman, V.K.; Kulkarni, B.D. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Appl. Math. Comput.* **2007**, *188*, 129–142. [CrossRef]
36. Alam, S.; Dobbie, G.; Riddle, P. An evolutionary particle swarm optimization algorithm for data clustering. In Proceedings of the 2008 IEEE Swarm Intelligence Symposium, St. Louis, MO, USA, 21–23 September 2008; pp. 1–6. [CrossRef]
37. Jain, M.; Saihjpal, V.; Singh, N.; Singh, S.B. An Overview of Variants and Advancements of PSO Algorithm. *Appl. Sci.* **2022**, *12*, 8392. [CrossRef]
38. Communications Security Establishment. Government of Canada. Available online: https://www.cse-cst.gc.ca/en (accessed on 2 Feburary 2023).
39. Canadian Institute for Cybersecurity. University of New Brunswick est.1785. Available online: https://www.unb.ca/cic/ (accessed on 2 Feburary 2023).
40. CSE-CIC-IDS2018 on AWS. Canadian Institute for Cybersecurity. Available online: https://www.unb.ca/cic/datasets/ids-2018.html (accessed on 2 Feburary 2023).
41. Registry of Open Data on AWS. A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018). Available online: https://registry.opendata.aws/cse-cic-ids2018 (accessed on 2 Feburary 2023).
42. TensorFlow. Distributed Training with TensorFlow. Available online: https://www.tensorflow.org/guide/distributed_training (accessed on 2 Feburary 2023).
43. Miranda, L.J. PySwarms: A research toolkit for Particle Swarm Optimization in Python. *J. Open Source Softw.* **2018**, *3*, 433. [CrossRef]
44. Haghighi, S.; Jasemi, M.; Hessabi, S.; Zolanvari, A. PyCM: Multiclass confusion matrix library in Python. *J. Open Source Softw.* **2018**, *3*, 729. [CrossRef]
45. Sammut, C.; Webb, G.I. *Encyclopedia of Machine Learning*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010. [CrossRef]
46. Sammut, C.; Webb, G.I. *Encyclopedia of Machine Learning and Data Mining*; Springer Publishing Company: Berlin/Heidelberg, Germany, 2017. [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.