applied sciences

MDPI

Article

Planet Optimization with Deep Convolutional Neural Network for Lightweight Intrusion Detection in Resource-Constrained IoT Networks

Khalid A. Alissa [1], Fatma S. Alrayes [2], Khaled Tarmissi [3], Ayman Yafoz [4], Raed Alsini [4], Omar Alghushairy [5], Mahmoud Othman [6] and Abdelwahed Motwakel [7,*]

1 SAUDI ARAMCO Cybersecurity Chair, Networks and Communications Department, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia
2 Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
3 Department of Computer Sciences, College of Computing and Information System, Umm Al-Qura University, P.O. Box 5555, Makkah 21955, Saudi Arabia
4 Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, P.O. Box 80200, Jeddah 21589, Saudi Arabia
5 Department of Information Systems and Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah 21589, Saudi Arabia
6 Department of Computer Science, Faculty of Computers and Information Technology, Future University in Egypt, New Cairo 11835, Egypt
7 Department of Computer and Self Development, Preparatory Year Deanship, Prince Sattam Bin Abdulaziz University, AlKharj 11942, Saudi Arabia
* Correspondence: a.ismaeil@psau.edu.sa

Abstract: Cyber security is becoming a challenging issue, because of the growth of the Internet of Things (IoT), in which an immense quantity of tiny smart gadgets push trillions of bytes of data over the Internet. Such gadgets have several security flaws, due to a lack of hardware security support and defense mechanisms, thus, making them prone to cyber-attacks. Moreover, IoT gateways present limited security features for identifying such threats, particularly the absence of intrusion detection techniques powered by deep learning (DL). Certainly, DL methods need higher computational power that exceeds the capability of such gateways. This article focuses on the development of Planet Optimization with a deep convolutional neural network for lightweight intrusion detection (PODCNN-LWID) in a resource-constrained IoT environment. The presented PODCNN-LWID technique primarily aims to identify and categorize intrusions. In the presented PODCNN-LWID model, two major processes are involved, namely, classification and parameter tuning. At the primary stage, the PODCNN-LWID technique applies a DCNN model for the intrusion identification process. Next, in the second stage, the PODCNN-LWID model utilizes the PO algorithm as a hyperparameter tuning process. The experimental validation of the PODCNN-LWID model is carried out on a benchmark dataset, and the results are assessed using varying measures. The comparison study reports the enhancements of the PODCNN-LWID model over other approaches.

Keywords: intrusion detection system; lightweight; deep learning; parameter tuning; security

Citation: A. Alissa, K.; S. Alrayes, F.; Tarmissi, K.; Yafoz, A.; Alsini, R.; Alghushairy, O.; Othman, M.; Motwakel, A. Planet Optimization with Deep Convolutional Neural Network for Lightweight Intrusion Detection in Resource-Constrained IoT Networks. *Appl. Sci.* **2022**, *12*, 8676. https://doi.org/10.3390/app12178676

Academic Editors: Howon Kim and Thi-Thu-Huong Le

Received: 5 August 2022
Accepted: 27 August 2022
Published: 30 August 2022

1. Introduction

The idea of the Internet of Things (IoT) depends upon the incorporation of recognizable, varied physical substances (instant cameras, humans, sensors, animals, vehicles, etc.) and the cyber world with the capability of transferring data in a network deprived of human-to-computer or human-to-human interfaces [1]. IoT applications may range from modest appliances for a smart home to complicated devices in a smart grid. The IoT offers a marvelous chance for humanity across the globe [2]. With various objectives, conflicting IoT

applications take a joint set of features. Generally, an IoT primary node can execute three different activities: data processing and utilization, data collection, and data transmission. Many approaches to enhancing data access, authentication, and confidentiality are stated in the literature; however, even with such systems, IoT networks are vulnerable to many assaults pointed at network disturbance [3]. The diversity, growth, ubiquity, and complexity of the IoT expands the possible assault surfaces. Thus, intrusion-preventing apparatuses and signature-related intrusion detection (ID) techniques are needed for the potential assaults in the IoT network [4]. An intrusion detection system (IDS) is a device or software application that monitors a network for abnormal activities or policy violations. Any intrusion activity or violation is normally reported either to an administrator, or gathered centrally through the use of security information and an event management system.

A defense system is needed for noticing new and effective intrusions. An IDS depends upon anomaly detection (i.e., statistically related) to achieve its purpose. Anomaly recognition has no need for previous recognition of assault signs [5]. Since IoT services cover an extensive range, there are varied transmission technologies with diverse values, which increases the threats to end-wise security. The assault surface size in the IoT is raised, due to the expansions in diversity and the complexity of the IoT [6]. Defense systems such as signature-related ID techniques do not function correctly for the adapted assaults, and, therefore, an IDS is necessary for new intrusions. The anomaly-related ID method drives this discovery structure, as there is no need for previous data regarding the assault signs [7]. Definite features of IoT systems were provided by several scholars in the facets of evolving an IDS.

The devices or nodes in the IoT are source-limited and work with less influence. It is impossible to host an orthodox IDS that needs higher computational abilities and a higher level of power [8]. The protocols utilized in the IoT system are not similar to the protocols utilized in traditional systems [9]. The protocols utilized in such networks, namely, IPv6 with Constrained Application Protocol (CoAP) and low-power, wireless personal area networks (6LoWPAN) make the network varied, leading to novel dimness and making it challenging to apply the IDS in the networks. Taking the aforementioned features into account, it becomes essential to advance a lightweight IDS that executes its purpose professionally in network security [10]. The word lightweight indicates that the scheme must operate on controlled sources obtainable in the IoT network nodes, and does not mean that the system is simple.

This article focuses on the development of Planet Optimization with a deep convolutional neural network for lightweight intrusion detection (PODCNN-LWID) in a resource-constrained IoT environment. The presented PODCNN-LWID technique primarily aims to identify and categorize intrusions. In the presented PODCNN-LWID model, two major processes are involved, namely, classification and parameter tuning. At the primary stage, the PODCNN-LWID technique applies the DCNN model for the intrusion identification process. Next, in the second stage, the PODCNN-LWID model utilizes the PO algorithm as a hyperparameter tuning process. The experimental validation of the PODCNN-LWID model is carried out on benchmark dataset, and the results are assessed using varying measures.

2. Related Works

Gali and Nidumolu [11] present a novel, chaotic bumble bee mating optimization (CBBMO) method for protected data communication, which has a trust-sensing method, called the CBBMOR–TSM method. For enhancing the convergence rates of the BBMO approach, the CBBMO approach is described as the incorporation of chaotic ideas into the traditional BBMO method. The purpose of the presented method is to devise a trust-sensing method and execute safe routing by utilizing the CBMO method. Mabayoje et al. [12] devise a multilevel dimensionality reduction structure, dependent on metaheuristic optimization and PCA. For attaining the research aim, PCA is implied for extracting features. GA and PSO methods, i.e., GA–PSO, are utilized to select features for choosing the discriminatory

attributes in order to advance the ID method. In the classifier phase, SVM and ANN techniques are employed for developing ID.

Kareem et al. [13] introduce a novel FS method by fostering the presentation of a Gorilla Troops Optimizer (GTO) related to the bird swarm algorithm (BSA). This BSA can be utilized for fostering act exploitation of GTO in the recently advanced GTO–BSA, due to a strong capability for finding possible areas for optimal solutions. In ref. [14], for the optimization of energy utilization, the most suitable cluster-head (CH) can be selected in the IoT. The presented study uses a hybrid metaheuristic method, such as WOA, which has simulated annealing (SA). For the selection of the optimum CH in groups of IoT systems, numerous presentation metrics such as the counting of alive residual energy, nodes, temperatures, cost function, and load are employed. Haddadpajouh et al. [15] devise a multi-kernel SVM for IoT cloud edge gateway malware-hunting, by utilizing the GWO method. This metaheuristic technique can be utilized for optimal feature selection, differentiating benign and malicious applications at the IoT cloud edge gateway. This method can be well-trained by the bytecode and opcode of IoT malware samples, and assessed utilizing the K-fold cross-validation method.

Li et al. [16] present an AI-related two-stage ID, authorized by software-based technology. It amenably stops network flows from a worldwide view, and identifies assaults intelligently. The authors use the Bat method for binary differential mutation, and swarm division for choosing typical features. After that, the authors use RF, by adaptively varying the sample weights by utilizing the biased voting system for classifying flows. Habib et al. [17] provide a novel technique to convert the old IDSs into multi-objective, smart, and evolutionary IDSs for IoT networks. Furthermore, this work offers an adapted system for IDSs that challenges the issue of selecting features. The adapted process stands on the incorporation of multi-objective PSO with the Lévy flight randomization component (MOPSO–Lévy).

3. The Proposed Lightweight IDS Model

In this article, a new PODCNN-LWID technique is developed for the recognition of intrusions in the IoT network. The presented PODCNN-LWID technique primarily aims to identify and categorize intrusions. In the presented PODCNN-LWID model, two major processes are involved, namely, classification and parameter tuning. Figure 1 depicts the overall process of the PODCNN-LWID approach. At the initial stage, the input networking data is preprocessed to fill missing values, using a median approach. Next, the preprocessed data is fed into the DCNN model to detect and classify intrusions. Finally, the PO algorithm is applied as a hyperparameter optimizer, to choose the hyperparameter values effectively.

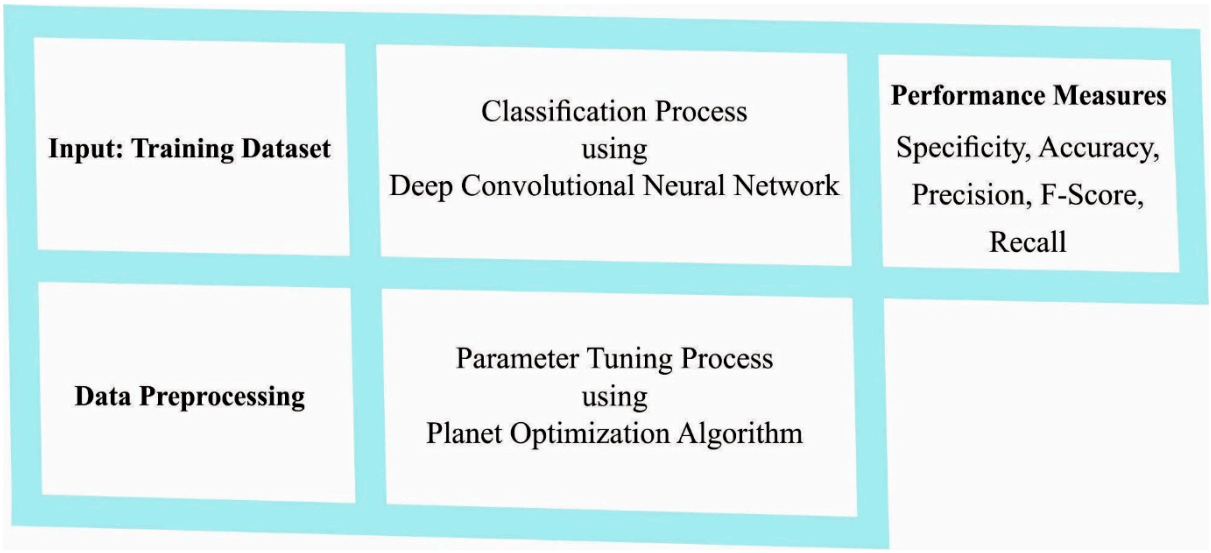

Figure 1. Overall process of PODCNN-LWID approach.

3.1. Stage I: Intrusion Detection Model

At the primary stage, the PODCNN-LWID technique applies the DCNN model for the intrusion identification process. CNNs are established with the concept of local connectivity. The spatial extension of all the connectivity is suggested as the receptive domain of nodes [18]. The local connectivity is accomplished by exchanging weight sums in the NN with Convs. In all the layers of the CNN, the input is Conv, with the weighted matrix (filter) to generate a feature map. The local connectivity and shared weight feature of CNNs decrease the entire amount of learnable parameters, resulting in further effective training. The deep CNN is generally separated into two important parts: the primary part is comprised of the order of two one-dimensional Conv blocks, with Conv1D layers of 32 and 64 channels in the first and second blocks, respectively, a Batchnorm layer, an ReLU activation function, and a max-pooling 1D layer. Other parts comprise the order of the fully connected (FC) layers. In two important Conv blocks, the input signal is encrypted by decreasing their length and improving the number of channels. Figure 2 showcases the infrastructure of the CNN.

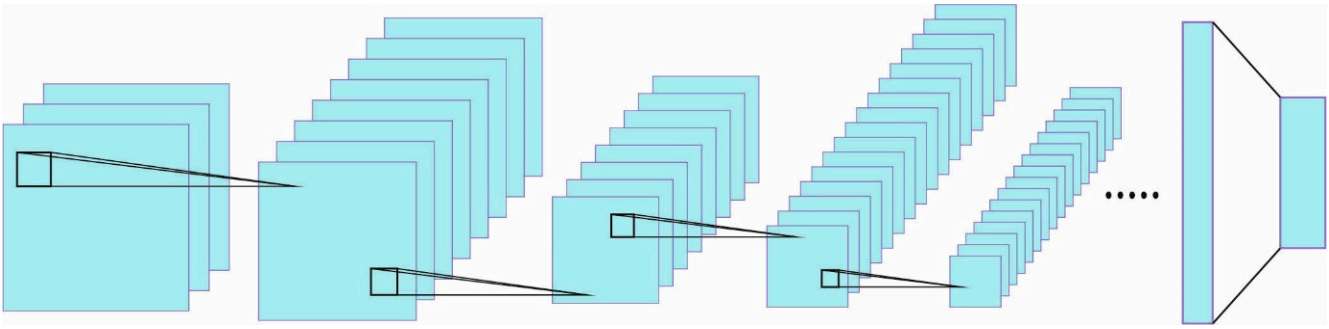

Figure 2. Structure of CNN.

The resultant second Conv block is concatenated, with input signals employing the remaining skip connections. However, it utilizes the network maintenance data as input at the deeper layers. Next, by concatenating the input signal and resultant Conv blocks, the FC layer is utilized for the final decision layer that creates the outcome. The resultant value of the Conv 1D layer with input size $(N,\ C_{in},\ L)$ and output $(N,\ C_{out},\ L_{out})$ is:

$$out\left(N_i,\ C_{out_j}\right) = bias\left(C_{out_j}\right) + \sum_{k=0}^{C_{in}-1} weight\left(C_{out_j},\ k\right) * input\ (N_i,\ k) \tag{1}$$

where $*$ refers to the valid cross-correlation function (in this work, it is the Conv function), N_i signifies the batch-size i^{th}, C_{outj} signifies the channel j^{th}, and L denotes the length of signal orders (when the input is image, width and height are employed rather than length). The length of resultant signal order is computed by utilizing the equation:

$$L_{out} = \left[\frac{L_{in} + 2 \times padding - dilation \times (kernelsize - 1) - 1}{stride} + 1\right] \tag{2}$$

where:

- stride implies the stride of cross-correlation;
- padding signifies the number of zero-paddings on both sides;
- dilation denotes the space amongst the kernel element;
- kernelsize stands for the size of the Conv kernel.

In order to achieve the max-pooling 1D, the resultant value with input size $(N,\ C,\ L)$ and outcome $(N,\ C,\ L_{out})$ is represented as:

$$out\left(N_j,\ C_{out_j}\right) = m = \max_{m=0,\ldots,kernel_{size\_1}} input(N_i,\ C_j,\ stride \times k + m) \tag{3}$$

in which N_i signifies the input i^{th}; C_j represents the channel j^{th}.

- kernelsize denotes the size of windows needed to obtain max over;
- stride indicates the stride of windows;
- padding demonstrates the number of zeros added on both sides;
- dilation represents the parameter that deals with the stride of elements from the window.

The length of the resultant signal order to the max-pooling 1D layer is computed utilizing the same equation from the ConvlD layer.

3.2. Stage II: Hyperparameter Tuning Model

In the second stage, the PODCNN-LWID model utilizes the PO algorithm as the hyperparameter tuning process. The universe is extremely big and has no boundaries, and large spaces are occupied with stars, planets, galaxies, and other stimulating astronomical phenomena [19]. For ease of visualization, and simplicity of the study, the solar system is used as representations. Initially, a structure comprised of the Moon, the Sun, and the Earth is taken into account in these cases. It is clear to everyone that the Sun maintains its' gravitational force to maintain the Earth revolving around it. The mass of the Sun is 330,000 times larger than that of the Earth. However, the Earth generates gravitation pull that is large enough to maintain the Moon's orbit around it. This illustrates the two factors that influence the movement of the planets, i.e., the mass, along with the distance between the two planets. Therefore, the process mimicking the law of gravitational force is shown below:

- The Sun acts as an optimum solution. In this problem, it has the highest mass, which implies that it has a great gravitational moment for the planets near and around it;
- Concerning the Sun (red planet) and other planets, there exists a gravitational pull between them. This moment can be determined by the mass and the distance between those objects. This implies that, even though the red planet has the biggest mass out of all the other planets, the moment on faraway planets is insignificant. This aids the process of preventing the local optima.

At the *t-th* iteration, the mass of the red planet is the largest, hence, it signifies the Sun. As the pink planet is closer to the red planet, it moves to the position of the red planet, due to gravitational pull $\left(M_p^t\right)$ between the planets and the red planet.

Nonetheless, the red planet at the *t-th* iteration does not have the preferred location that we are trying to find, namely, the minimal optima. In other words, if all the planets move towards the red planet, the procedure becomes trapped in the local optima. On the other hand, the blue planet is in a prospective position and is farther from the red planet. The communication of the red planet with the blue planet $\left(M_b^t\right)$ is smaller, as it is farther from the red planet at the *t-th* iteration. Therefore, it is wide open for the blue planet to seek the best position in the following iteration.

The primary objective depends on the abovementioned two concepts. In addition, the red planet is the accurate objective of examination, and obviously, we do not have its' correct position. In such cases, the planet with the greatest mass at the *t-th* iteration acts as the Sun simultaneously.

The application of the procedure is shown below:

Stage 1: Initially, the best process is the one wherein the last finest solution is autonomous of the primary locations. Nonetheless, realism conflicts with most of the stochastic approaches. When the target area is mountainous, and the global optimal is positioned in a remote minor region, an early population has significance. When an early arbitrary population does not make solution in the surrounding area of the global-searching phase of the actual population, then the possibility that the population focuses on the correct optimal could be lower.

On the other hand, with the construction of a primary solution near to the global optimum location, the possibility of the convergence of the population to the optimum position becomes higher. Indeed, globalization is comparatively large, and subsequently,

the initial population performs a major part. Preferably, the introduction must utilize crucial sampling methods, for example, the technique employed in the Monte Carlo model for sampling the solution. Although this request possess sufficient intelligence from the algorithm, it could not be fulfilled by almost all of the processes.

Similar to selecting the early population, selecting the finest solution in a new population in the part of the red planet regarding other planets' stirring towards the position is significant. This process determines the accuracy and the convergence rate of the problem.

Consequently, the initial step is to discover a better solution for playing the role of the optimum solution, in order to raise the accuracy and convergence of the algorithm in the initial iteration.

Stages 2: M factor.

$$M = \left|\vec{F}\right| R_{ij} = G\frac{m_i m_j}{R_{ij}^2} \times R_{ij} \tag{4}$$

Here, the subsequent parameter can be determined:

- The mass of the planet:

$$m_i,\ m_j = \frac{1}{a^{obj_{ij}/\alpha}} \tag{5}$$

in which $a = 2$ represents a constant variable, and $\alpha = |\max(obj) - obj_{sun}|$. This implies that when the value of an objective function of a planet is small, the mass of a planet is large. obj_{ij}, max (obj), and obj_{sun} show the objective function value of *i-th* or *j-th* planets, e.g., the worst planet and the red planet, respectively;

- The distance between the *i-th* and *j-th* objects with "Dim" as dimension, Cartesian distance, is evaluated as follows:

$$R_{ij} = \|X_i^t - X_j^t\| = \sqrt{\sum_{k=1}^{Dim} \left(X_i^t - X_j^t\right)^2} \tag{6}$$

- G indicates a variable, and it is equivalent to unity.

Stage 3: Global search. A formulation constructed to mimic global search is as follows

$$\overrightarrow{X_i^{t+1}} = \vec{X_i^t} + b \times \beta \times r_1 \times \left(\overrightarrow{X_{Sun}^t} - \vec{X_i^t}\right) \tag{7}$$

The left side of the equation exemplifies the existing location of the planet *i-th* in the $(t+1)$ iteration, whereas the right side contains the foremost components in the following:

- $\vec{X_i^t}$ shows the existing location of planet *i-th* at the *t-th* iteration;
- $\beta = M_i^t / M_{mu}^t, r_1 = rand(0,1), b$ indicates a constant variable;
- X_{Sun}^r indicates the existing location of the red planet in the *t-th* iteration.

The expression β indicates a coefficient that can be determined by M, M_i^t refers to the gravity of the red planet *i-th* at the *t-th* iteration, and M_m^t indicates the max $\left(M_i^t\right)$ value at the *t-th* iteration. Consequently, the β coefficient has a value within (0, 1).

Stage 4: Local search. In this phase, the accurate position is often the preferred objective to be established. This aim is easy or difficult to accomplish, based on how complex the problem is. In all instances, there is only potential to find an estimated value that fits the actual condition. Specifically, the correct Sun position is in the space between the solution.

Remarkably, while Jupiter is the largest planet in the solar system, Mercury is the planet whose position is closer to the red planet. This implies that the optimum solution location to the accurate position of the Sun at the *t-th* iteration might not be nearer than the position of other solutions to the accurate Sun position.

Once the distance between the planets and the Sun is smaller, the local search progression is implemented. From the abovementioned method, the planet with the highest mass operates as the red planet, and in such cases, it is Jupiter that is nearer to the red planet, and moves to the position of the Sun. In other words, the planet moves a smaller distance between its' own position and the red planet at the *t-th* iteration, rather than moving directly towards the red planet. This step aims to improve the performance in a narrow region of search space, and it is given below:

$$\overrightarrow{X_i^{t+1}} = \vec{X_i^t} + c \times r_1 \times \left(r_2 \times \overrightarrow{X_{sun}^t} - \vec{X_i^t}\right) \tag{8}$$

In Equation (8), $c = c_0 - t/T$, t refers to the *t-th* iteration, T shows the maximal iteration count, and $c_0 = 2.r_2$ indicates the Gaussian distribution function as follows:

$$f(x,\ \mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\sigma)^2}{2\sigma^2}\right) \tag{9}$$

Several evolutionary procedures are also randomized through stochastic processes, namely, Lévy distribution and power-law distribution. Normal or Gaussian distribution is the most common, because the maximum number of physical parameters involving uncertainty or errors in the measurement, light intensity, etc., obeys these distributions. The PO algorithm makes a derivation of a fitness function (FF), which results in an enhanced classifier performance. In this article, the reduction in the classifier error rate can be regarded as the FF, as presented in Equation (10).

$$fitness\ (x_i) = \frac{number\ of\ misclassified\ samples}{Total\ number\ of\ samples} * 100 \tag{10}$$

4. Results and Discussion

The proposed model is simulated using Python 3.6.5 tool on a PC i5-8600k, with GeForce 1050 Ti 4 GB, 16 GB RAM, 250 GB SSD, and 1 TB HDD. As the DCNN model has parameters, the hyperparameter tuning process is performed. The learning rate is kept as 0.01. The number of filters is 32 in the primary CNN layer, 64 in the succeeding CNN layer, and 128 in the last CNN layer. The parameter max-pooling length is set to 2 in all the max-pooling layers, and dropout to 0.01. When the number of CNN layers increases from three to four, the performance is reduced and, therefore, a three-level CNN is used. Lastly, two dense layers are included along with the CNN layer; the first dense layer is composed of 512 neurons and the second layer is composed of 128 neurons. These layers use ReLU as the activation function.

The experimental validation of the PODCNN-LWID model is tested using a dataset comprising 5500 samples with 11 classes, as depicted in Table 1. We used the CICIDS2017 dataset, which is comprised of normal and the most up-to-date common attacks, and resembles the true, real-world data (PCAPs). The dataset is available at https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 13 June 2022). It encompasses the network traffic analyses results from the use of CICFlowMeter, with labeled flows based on the time stamp, source and destination IPs, source and destination ports, protocols, and attacks (CSV files). This dataset was gathered for five continuous days (Monday–Friday), with various attacks as well as normal information. The proposed model is simulated using the Python tool. The set of measures used to examine the results are accuracy, precision, recall, specificity, and *F-score*.

Figure 3 illustrates the confusion matrix offered by the PODCNN-LWID model with the entire dataset. The PODCNN-LWID model identifies 493 samples in class 1; 481 samples in class 2; 490 samples in class 3; 484 samples in class 4; 483 samples in class 5; 485 samples in class 6; 490 samples in class 7; 484 samples in class 8; 470 samples in class 9; 493 samples in class 10; and 492 samples in class 11.

Table 1. Dataset details.

Label	Class	No. of Instances
1	Normal	500
2	Botnet	500
3	DoSSlowhttptest	500
4	FTP-Patator	500
5	SSH-Patator	500
6	DoSGoldenEye	500
7	DoSslowloris	500
8	Heartbleed	500
9	PortScan	500
10	DDoS	500
11	DoSHulk	500
Total No. of Instances		**5500**

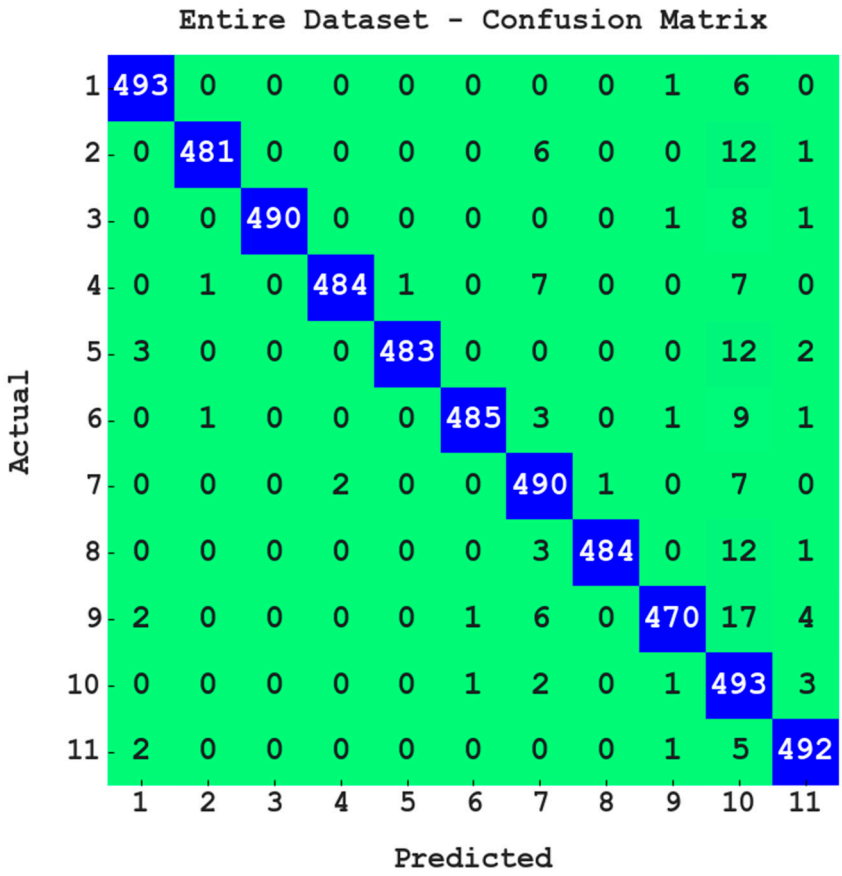

Figure 3. Confusion matrix of PODCNN-LWID approach with entire dataset.

Table 2 reports the intrusion classification output of the PODCNN-LWID model with the entire dataset. In class 1, the PODCNN-LWID model provides $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.75%, 98.60%, 98.60%, 99.86%, and 98.60%, respectively. In the meantime, in class 2, the PODCNN-LWID approach offers $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.62%, 99.59%, 96.20%, 99.96%, and 97.86%, respectively. Likewise, in class 3, the PODCNN-LWID technique presents $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.82%, 100%, 98%, 100%, and 98.99%, respectively. Moreover, in class 4, the PODCNN-LWID algorithm renders $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.67%, 99.59%, 96.80%, 99.96%, and 98.17%, respectively. Finally, in class 5, the PODCNN-LWID approach grants $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.67%, 99.79%, 96.60%, 99.98%, and 98.17%, respectively.

Table 2. Result analysis of PODCNN-LWID approach with distinct class labels for entire dataset.

Entire Dataset					
Labels	**Accuracy**	**Precision**	**Recall**	**Specificity**	***F-Score***
1	99.75	98.60	98.60	99.86	98.60
2	99.62	99.59	96.20	99.96	97.86
3	99.82	100.00	98.00	100.00	98.99
4	99.67	99.59	96.80	99.96	98.17
5	99.67	99.79	96.60	99.98	98.17
6	99.69	99.59	97.00	99.96	98.28
7	99.33	94.78	98.00	99.46	96.36
8	99.69	99.79	96.80	99.98	98.27
9	99.36	98.95	94.00	99.90	96.41
10	98.15	83.84	98.60	98.10	90.62
11	99.62	97.43	98.40	99.74	97.91
Average	**99.49**	**97.45**	**97.18**	**99.72**	**97.24**

Figure 4 demonstrates the confusion matrix offered by the PODCNN-LWID technique for 70% of the TR dataset. The PODCNN-LWID algorithm identifies 344 samples in class 1; 340 samples in class 2; 350 samples in class 3; 341 samples in class 4; 339 samples in class 5; 336 samples in class 6; 351 samples in class 7; 338 samples in class 8; 322 samples in class 9; 335 samples in class 10; and 338 samples in class 11.

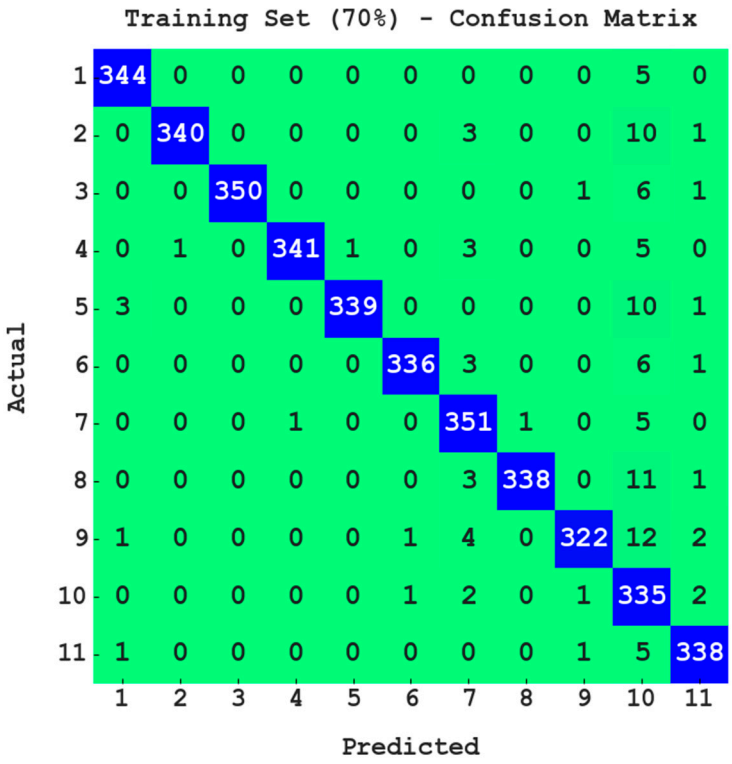

Figure 4. Confusion matrix of PODCNN-LWID approach with 70% of TR data.

Table 3 portrays the intrusion classification output of the PODCNN-LWID technique for 70% of the TR dataset. For class 1, the PODCNN-LWID algorithm offers $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.74%, 98.57%, 98.57%, 99.86%, and 98.57%, respectively. In the meantime, for class 2, the PODCNN-LWID approach grants $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.61%, 99.71%, 96.05%, 99.97%, and 97.84%, respectively. In addition, for class 3, the PODCNN-LWID algorithm renders $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.79%, 100%, 97.77%, 100%, and 98.87%, respectively. Additionally, for class 4, the PODCNN-LWID technique offers $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.71%,

99.71%, 97.15%, 99.97%, and 98.41%, respectively. Lastly, for class 5, the PODCNN-LWID approach renders $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.61%, 99.71%, 96.03%, 99.97%, and 97.84%, respectively.

Table 3. Result analysis of PODCNN-LWID approach with distinct class labels with 70% of TR data.

Training Set (70%)					
Labels	**Accuracy**	**Precision**	**Recall**	**Specificity**	***F-Score***
1	99.74	98.57	98.57	99.86	98.57
2	99.61	99.71	96.05	99.97	97.84
3	99.79	100.00	97.77	100.00	98.87
4	99.71	99.71	97.15	99.97	98.41
5	99.61	99.71	96.03	99.97	97.84
6	99.69	99.41	97.11	99.94	98.25
7	99.35	95.12	98.04	99.48	96.56
8	99.58	99.71	95.75	99.97	97.69
9	99.40	99.08	94.15	99.91	96.55
10	97.90	81.71	98.24	97.86	89.21
11	99.58	97.41	97.97	99.74	97.69
Average	**99.45**	**97.28**	**96.98**	**99.70**	**97.04**

Figure 5 exemplifies the confusion matrix presented by the PODCNN-LWID approach with 30% of the TS data. The PODCNN-LWID technique identifies 149 samples in class 1; 141 samples in class 2; 140 samples in class 3; 143 samples in class 4; 144 samples in class 5; 149 samples in class 6; 139 samples in class 7; 146 samples in class 8; 148 samples in class 9; 158 samples in class 10; and 154 samples in class 11.

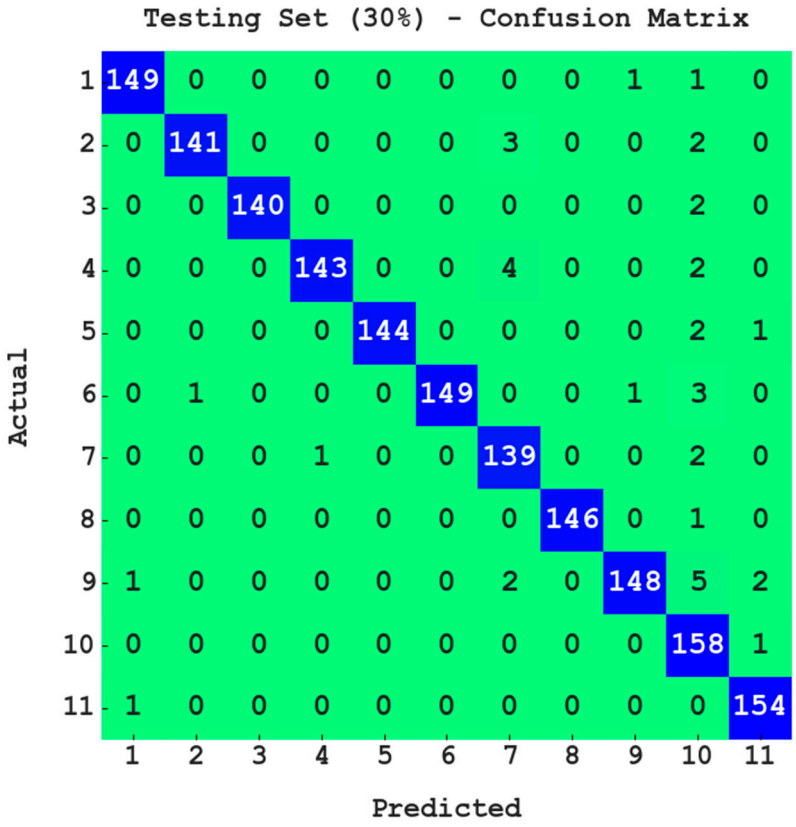

Figure 5. Confusion matrix of PODCNN-LWID approach with 30% of TS data.

Table 4 shows the intrusion classification output of the PODCNN-LWID approach with 30% of the TS dataset. For class 1, the PODCNN-LWID methodology presents $accu_y$, $prec_n$,

$reca_l$, $spec_y$, and F_{score} values of 99.76%, 98.68%, 98.68%, 99.87%, and 98.68%, respectively. In the meantime, for class 2, the PODCNN-LWID algorithm renders $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.64%, 99.30%, 96.58%, 99.93%, and 97.92%, respectively. Similarly, for class 3, the PODCNN-LWID technique produces $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.88%, 100%, 98.59%, 100%, and 99.29% respectively. Further, for class 4, the PODCNN-LWID technique offer $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.58%, 99.31%, 95.97%, 99.93%, and 97.61%, respectively. Finally, for class 5, the PODCNN-LWID approach produces $accu_y$, $prec_n$, $reca_l$, $spec_y$, and F_{score} values of 99.82%, 100%, 97.96%, 100%, and 98.97%, respectively.

Table 4. Result analysis of PODCNN-LWID approach with distinct class labels with 30% of TS data.

	Testing Set (30%)				
Labels	**Accuracy**	**Precision**	**Recall**	**Specificity**	***F-Score***
1	99.76	98.68	98.68	99.87	98.68
2	99.64	99.30	96.58	99.93	97.92
3	99.88	100.00	98.59	100.00	99.29
4	99.58	99.31	95.97	99.93	97.61
5	99.82	100.00	97.96	100.00	98.97
6	99.70	100.00	96.75	100.00	98.35
7	99.27	93.92	97.89	99.40	95.86
8	99.94	100.00	99.32	100.00	99.66
9	99.27	98.67	93.67	99.87	96.10
10	98.73	88.76	99.37	98.66	93.77
11	99.70	97.47	99.35	99.73	98.40
Average	**99.57**	**97.83**	**97.65**	**99.76**	**97.69**

The training accuracy (TRA) and validation accuracy (VLA) gained by the PODCNN-LWID approach with the test dataset is shown in Figure 6. The experimental outcome implies that the PODCNN-LWID methodology attains maximal TRA and VLA values. The VLA is greater than the TRA.

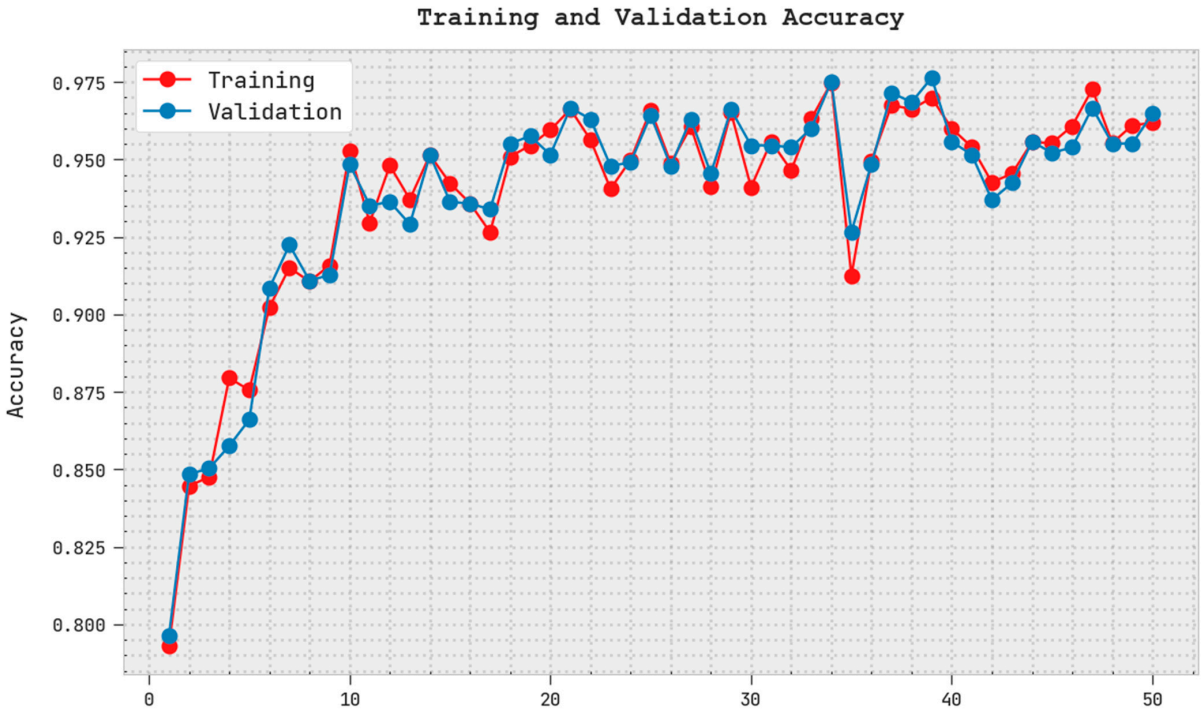

Figure 6. TRA and VLA analysis of PODCNN-LWID approach.

The training loss (TRL) and validation loss (VLL) obtained by the PODCNN-LWID algorithm with the test dataset are displayed in Figure 7. The experimental outcome indicates that the PODCNN-LWID method exhibits minimal TRL and VLL values. In particular, the VLL is less than the TRL.

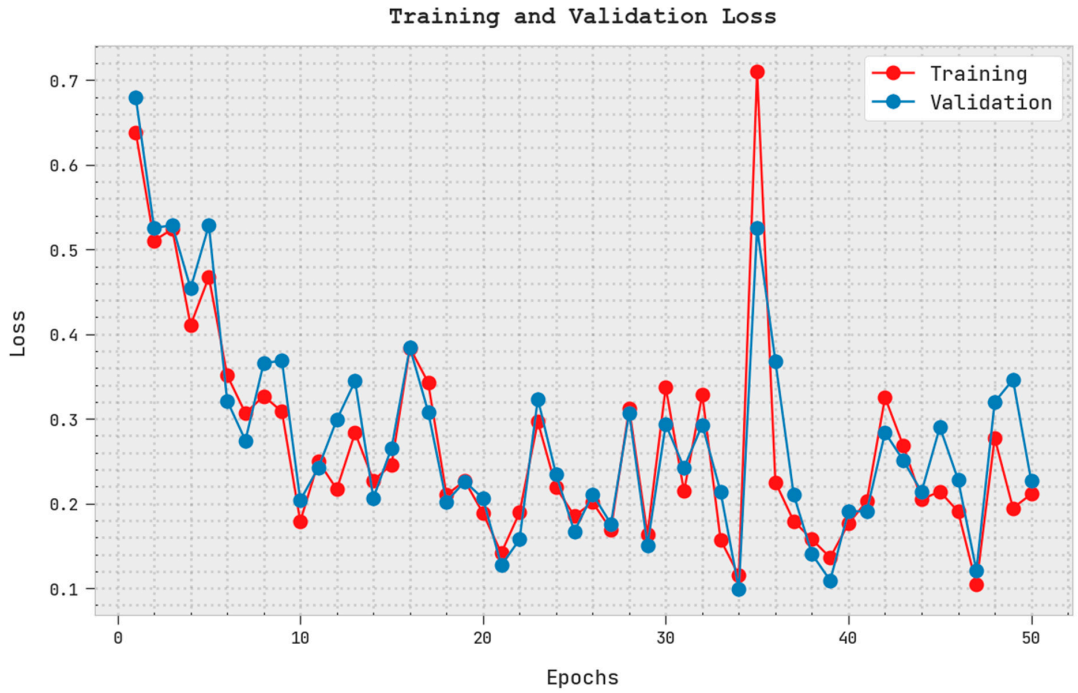

Figure 7. TRL and VLL analysis of PODCNN-LWID approach.

A clear precision–recall inspection of the PODCNN-LWID algorithm with the test dataset is portrayed in Figure 8. The figure denotes that the PODCNN-LWID approach results in enhanced precision–recall values under all classes.

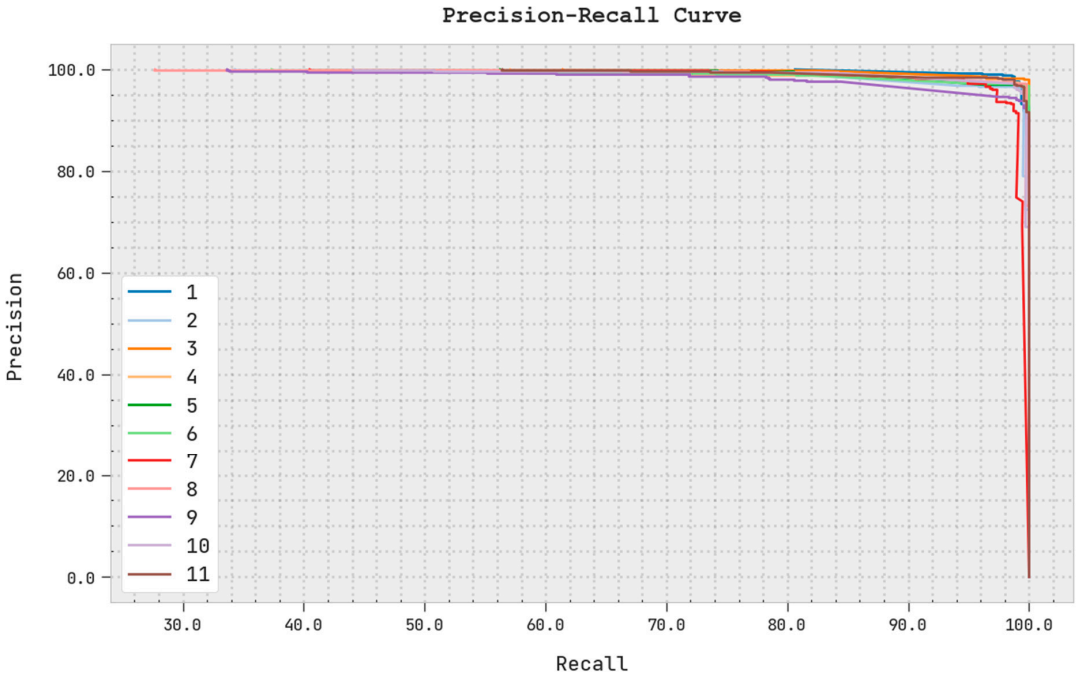

Figure 8. Precision–recall analysis of PODCNN-LWID approach.

A brief ROC investigation of the PODCNN-LWID algorithm on the test dataset is displayed in Figure 9. The results indicate that the PODCNN-LWID method exhibits its ability in categorizing distinct classes in the test dataset.

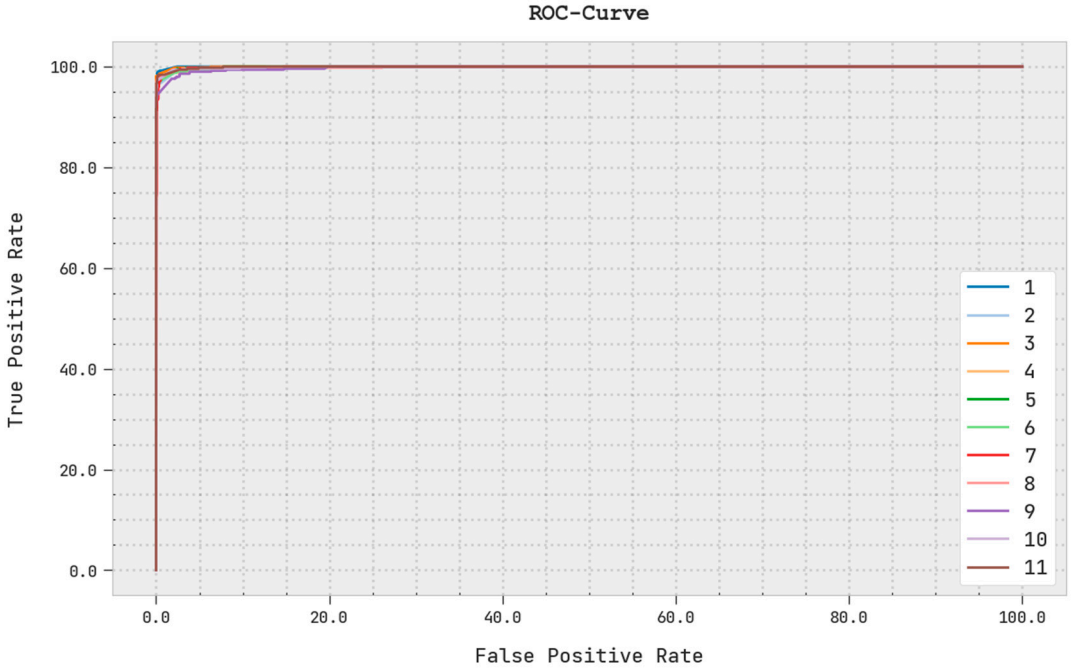

Figure 9. ROC analysis of PODCNN-LWID approach.

Table 5 highlights the intrusion detection efficacy of the PODCNN-LWID model in terms of distinct measures. The experimental results indicate that the PODCNN-LWID model reports enhanced results. For instance, based on $accu_y$, the PODCNN-LWID model offers an increased $accu_y$ of 99.57%, whereas the CNN-MCL, XGB, RF, SVC, ANN, and E-ML models accomplish decreased $accu_y$ values of 99.09%, 99.15%, 99.02%, 99.26%, 99.12%, and 99.48%, respectively. In contrast, based on $sens_y$, the PODCNN-LWID approach renders an increased $sens_y$ of 97.65%, whereas the CNN-MCL, XGB, RF, SVC, ANN, and E-ML algorithms establish decreased $sens_y$ values of 97.19%, 96.89%, 96.06%, 96.21%, 97.01%, and 96.20%, respectively. Finally, based on $spec_y$, the PODCNN-LWID technique has an increased $spec_y$ of 99.76%, whereas the CNN-MCL, XGB, RF, SVC, ANN, and E-ML approaches accomplish decreased $spec_y$ values of 99%, 99.23%, 99.60%, 99.11%, 99.21%, and 99.31% respectively. From the results and discussion, it is apparent that the PODCNN-LWID model accomplishes the maximum intrusion detection performance, with an accuracy of 99.57%. The enhanced performance of the PODCNN-LWID model is due to the inclusion of the PO-algorithm-based hyperparameter tuning process.

Table 5. Comparative analysis of PODCNN-LWID approach with existing algorithms.

Methods	Accuracy	Sensitivity	Specificity
PODCNN-LWID	99.57	97.65	99.76
CNN-MCL	99.09	97.19	99.00
XGB	99.15	96.89	99.23
RF	99.02	96.06	99.60
SVC	99.26	96.21	99.11
ANN	99.12	97.01	99.21
E-ML	99.48	96.20	99.31

5. Conclusions

In this article, a new PODCNN-LWID technique is developed for the recognition of intrusions in the IoT network. The presented PODCNN-LWID technique primarily aims to identify and categorize intrusions. In the presented PODCNN-LWID model, two major processes are involved, namely, classification and parameter tuning. At the primary stage, the PODCNN-LWID technique applies the DCNN model for the intrusion identification process. Next, in the second stage, the PODCNN-LWID model utilizes the PO algorithm as a hyperparameter tuning process. The experimental validation of the PODCNN-LWID model is carried out on a benchmark dataset, and the results are assessed using varying measures. The comparison study reports the enhancements of the PODCNN-LWID model over other approaches. In the future, feature selection and outlier removal processes can be integrated to boost the efficiency of the projected approach.

Author Contributions: Conceptualization, K.A.A.; Data curation, F.S.A.; Formal analysis, F.S.A.; Funding acquisition, A.M.; Investigation, K.T.; Methodology, K.A.A. and K.T.; Project administration, A.Y. and A.M.; Resources, A.Y.; Software, R.A. and O.A.; Supervision, O.A.; Validation, M.O.; Visualization, M.O.; Writing—original draft, K.A.A.; Writing—review & editing, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R319), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant Code: (22UQU4331004DSR02).

Institutional Review Board Statement: This article does not contain any studies with human participants performed by any of the authors.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable to this article as no datasets were generated during the current study.

Conflicts of Interest: The authors declare that they have no conflict of interest. The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript.

References

1. Naik, B.; Obaidat, M.S.; Nayak, J.; Pelusi, D.; Vijayakumar, P.; Islam, S.H. Intelligent Secure Ecosystem Based on Metaheuristic and Functional Link Neural Network for Edge of Things. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1947–1956. [CrossRef]
2. Revanesh, M.; Sridhar, V. A trusted distributed routing scheme for wireless sensor networks using blockchain and meta-heuristics-based deep learning technique. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4259. [CrossRef]
3. Shakil, M.; Mohammed, A.F.Y.; Arul, R.; Bashir, A.K.; Choi, J.K. A novel dynamic framework to detect DDoS in SDN using metaheuristic clustering. *Trans. Emerg. Telecommun. Technol.* **2019**, *33*, e3622. [CrossRef]
4. Murali Mohan, V.; Balajee, R.M.; Hiren, K.M.; Rajakumar, B.R.; Binu, D. Hybrid machine learning approach based intrusion detection in cloud: A metaheuristic assisted model. *Multiagent Grid Syst.* **2022**, *18*, 21–43. [CrossRef]
5. Srivastava, A.; Mishra, P.K. A Survey on WSN Issues with its Heuristics and Meta-Heuristics Solutions. *Wirel. Pers. Commun.* **2021**, *121*, 745–814. [CrossRef]
6. Jaber, M.M.; Alameri, T.; Ali, M.H.; Alsyouf, A.; Al-Bsheish, M.; Aldhmadi, B.K.; Ali, S.Y.; Abd, S.K.; Ali, S.M.; Albaker, W.; et al. Remotely monitoring COVID-19 patient health condition using metaheuristics convolute networks from IoT-based wearable device health data. *Sensors* **2022**, *22*, 1205. [CrossRef]
7. Al-Qarafi, A.; Alrowais, F.; Alotaibi, S.S.; Nemri, N.; Al-Wesabi, F.N.; Al Duhayyim, M.; Marzouk, R.; Othman, M.; Al-Shabi, M. Optimal Machine Learning Based Privacy Preserving Blockchain Assisted Internet of Things with Smart Cities Environment. *Appl. Sci.* **2022**, *12*, 5893. [CrossRef]
8. Cho, H.H.; Wu, H.T.; Lai, C.F.; Shih, T.K.; Tseng, F.H. Intelligent charging path planning for IoT network over Block-chain-based edge architecture. *IEEE Internet Things J.* **2020**, *8*, 2379–2394. [CrossRef]
9. Pajooh, H.; Rashid, M.; Alam, F.; Demidenko, S. Multi-Layer Blockchain-Baseds Security Architecture for Internet of Things. *Sensors* **2021**, *21*, 772. [CrossRef] [PubMed]
10. Chen, C.Y.; Cho, H.H.; Tsai, M.Y.; Hann, A.S.H.; Chao, H.C. Detecting LDoS in NB-IoTs by using metaheuristic-based CNN. *Int. J. Ad Hoc Ubiquitous Comput.* **2021**, *37*, 74–84. [CrossRef]

11. Gali, S.; Nidumolu, V. An intelligent trust sensing scheme with metaheuristic based secure routing protocol for Internet of Things. *Clust. Comput.* **2021**, *25*, 1779–1789. [CrossRef]
12. Mabayoje, M.A.; Adewole, K.S.; Ekeruvwe, O.E.; Ajao, J.F.; Akinrotimi, A.O.; Balogun, A.O. A Metaheuristic Approach to Network Intrusion Detection. *Ilorin J. Comput. Sci. Inf. Technol.* **2022**, *5*, 22–34.
13. Kareem, S.S.; Mostafa, R.R.; Hashim, F.A.; El-Bakry, H.M. An Effective Feature Selection Model Using Hybrid Metaheuristic Algorithms for IoT Intrusion Detection. *Sensors* **2022**, *22*, 1396. [CrossRef] [PubMed]
14. Iwendi, C.; Maddikunta, P.K.R.; Gadekallu, T.R.; Lakshmanna, K.; Bashir, A.K.; Piran, J. A metaheuristic optimization approach for energy efficiency in the IoT networks. *Softw. Pract. Exp.* **2020**, *51*, 2558–2571. [CrossRef]
15. Haddadpajouh, H.; Mohtadi, A.; Dehghantanaha, A.; Karimipour, H.; Lin, X.; Choo, K.-K.R. A Multikernel and Metaheuristic Feature Selection Approach for IoT Malware Threat Hunting in the Edge Layer. *IEEE Internet Things J.* **2020**, *8*, 4540–4547. [CrossRef]
16. Li, J.; Zhao, Z.; Li, R.; Zhang, H. AI-Based Two-Stage Intrusion Detection for Software Defined IoT Networks. *IEEE Internet Things J.* **2018**, *6*, 2093–2102. [CrossRef]
17. Habib, M.; Aljarah, I.; Faris, H. A Modified Multi-objective Particle Swarm Optimizer-Based Lévy Flight: An Approach Toward Intrusion Detection in Internet of Things. *Arab. J. Sci. Eng.* **2020**, *45*, 6081–6108. [CrossRef]
18. Van, S.P.; Le, H.M.; Thanh, D.V.; Dang, T.D.; Loc, H.H.; Anh, D.T. Deep learning convolutional neural network in rainfall–runoff modelling. *J. Hydroinform.* **2020**, *22*, 541–561. [CrossRef]
19. Sang-To, T.; Hoang-Le, M.; Khatir, S.; Mirjalili, S.; Wahab, M.A.; Cuong-Le, T. Forecasting of excavation problems for high-rise building in Vietnam using planet optimization algorithm. *Sci. Rep.* **2021**, *11*, 23809. [CrossRef] [PubMed]