Article

IoT Botnet Attack Detection Model Based on DBO-Catboost

Changjin Yang ¹, Weili Guan ²*, and Zhijie Fang ³

¹ School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China; 2013391076@st.gxu.edu.cn
² College of Digital Economics, Nanning University, Nanning 530299, China
³ College of Electrical Engineering, Guangxi University of Science and Technology, Liuzhou 545006, China; nnfang@semi.ac.cn
* Correspondence: 2113391032@st.gxu.edu.cn

Abstract: With the widespread adoption of Internet of Things (IoT) technology, the increasing number of IoT devices has led to a rise in serious network security issues. Botnets, a major threat in network security, have garnered significant attention over the past decade. However, detecting these rapidly evolving botnets remains a challenge, with current detection accuracy being relatively low. Therefore, this study focuses on designing efficient botnet detection models to enhance detection performance. This paper improves the initial population generation strategy of the Dung Beetle Optimizer (DBO) by using the centroid opposition-based learning strategy instead of the original random generation strategy. The improved DBO is applied to optimize Catboost parameters and is employed in the field of IoT botnet detection. Performance comparison experiments are conducted using real-world IoT traffic datasets. The experimental results demonstrate that the proposed method outperforms other models in terms of accuracy and F1 score, indicating the effectiveness of the proposed approach in this field.

Keywords: IoT; botnet; catboost; dung beetle optimizer

1. Introduction

The rapid development of the Internet of Things (IoT) has brought many conveniences to modern society, such as smart homes [1], smart grids [2], smart cities [3], and the Industrial Internet of Things [4]. The IoT enables communication between machines through the internet, connecting a large number of online devices and allowing them to actively participate in the network. The decreasing prices of IoT devices are accelerating their proliferation and growth. According to Cisco's prediction [5], by the end of 2023 the global number of IoT devices will exceed 29 billion. The exponential growth of IoT devices is widely applied in various applications to enhance service quality and efficiency. However, IoT devices face limitations due to resource constraints and the need to adapt to heterogeneous environments, which introduce security vulnerabilities.

On one hand, a large number of commonly used devices such as televisions, refrigerators, surveillance cameras, doors, windows, cups, and light bulbs are interconnected with other devices through technology such as Bluetooth and ZigBee, greatly facilitating people's lives. However, this also provides attackers with more opportunities to invade and steal user information, including privacy and property security [6]. On the other hand, the IoT is an extension and expansion of the traditional Internet, making the core network more complex and massive. The deployment of numerous outdoor and resource-constrained sensing nodes makes the network more vulnerable to external attacks. If the network experiences a failure, the consequences can be even more severe [7]. When IoT devices suffer from unauthorized intrusions, this not only involves intangible information assets but also tangible objects in real life. The target devices being compromised can lead to user information leakage, property loss, and even threats to users' personal safety.
indicate a corresponding increase in network attacks as the number of these insecure IoT devices rapidly grows.

One common attack scenario is when network hackers target IoT devices to turn them into part of a zombie network. Once infected, these devices are controlled and utilized by attackers to expand the zombie network continuously. After gaining control over multiple IoT devices, the infected zombie network devices are used to carry out various attacks, such as gaining unauthorized access to user information, massive spam email sending, and executing DDoS attacks. The impact of IoT zombie networks is extremely severe, as exemplified by the Mirai botnet in 2016, which infected millions of devices and conducted the largest DDoS attack in history [8,9]. As mentioned earlier, the creation of IoT botnet networks is a major attack method. Therefore, people usually employ various security control measures, such as intrusion detection, to detect and prevent IoT botnet attacks. Although these methods are effective to some extent, they cannot detect the formation of zero-day IoT botnet networks without known signatures [10]. In addition, infected IoT devices within a zombie network may not exhibit obvious signs of infection and can continue to operate normally without detection. Therefore, detecting and identifying infected IoT devices is a challenging task [11]. To protect IoT devices from botnet attacks, it has been recognized that an effective zombie network detection model needs to be established for defense [12]. Currently, research on IoT zombie network detection methods can be broadly classified into two categories: machine-learning-based detection methods and deep-learning-based detection methods. These detection methods have made significant progress in terms of detection accuracy. However, there are still limitations, including: (1) deep-learning-based methods demanding high computational resources, which are not suitable for resource-constrained IoT devices; (2) overly complex machine learning models; and (3) machine learning detection models being limited to the specific datasets they were trained on.

This paper aims to propose an IoT botnet detection model based on DBO-Catboost. In this model, the DBO algorithm is improved by incorporating a centroid opposition-based learning strategy to generate the initial population. The improved DBO algorithm produces initial solutions that are closer to the algorithm’s optimal solution. The improved DBO algorithm is then used to optimize the hyperparameters of Catboost, and applied to the IoT botnet detection. The experiments utilize real IoT datasets, Botnet and Bot-IoT, and make comparisons with existing detection models. The experimental results demonstrate that the IoT botnet detection model based on IDBO-Catboost outperforms other state-of-the-art classifiers in terms of accuracy and F1 score. It also exhibits higher detection efficiency and possesses a certain degree of generalization capability.

2. Related Works

With the continuous development of machine learning, machine learning algorithms have been widely applied in the field of IoT security and play a crucial role in IoT zombie network detection.

Ryu et al. [13] conducted research on different ML algorithms such as Naive Bayes, Decision Trees, etc., to detect zombie network behavior in network traffic. The researchers evaluated these methods using the CTU-13 zombie network dataset, and most of the approaches performed well, especially ensemble algorithms that outperformed individual classifiers in detecting zombie network attacks in network traffic. Xiao et al. [14] compared the random forest algorithm with five other machine learning algorithms as training models. The experimental results showed that the proposed model performed the best according to the given evaluation criteria. Pei et al. [15] presented a zombie network detection model based on Light GBM, and the experimental results demonstrated that the model achieved a higher detection accuracy and effectively detected zombie networks. Alshamkhany et al. [16] proposed the use of emerging machine learning techniques for detecting zombie networks or malicious traffic activities. Four classifiers, namely Naive Bayes, K-Nearest Neighbors, Support Vector Machines, and Decision Trees, were applied,
and the experimental results showed that the decision tree model outperformed the other classifier models. Biradar et al. [17] employed a combination of supervised machine learning techniques for zombie network detection, using DNS data in the detection process. The main advantage of this method was its superior accuracy in detecting zombie networks, but the most significant drawback was the complexity of the model. Injadat et al. [18] presented a machine-learning-based optimization framework that combined Bayesian optimization with Gaussian process algorithms and decision tree classification models to detect attacks on IoT devices in an effective and efficient manner. Vishwakarma [19] employed techniques such as random forest, decision trees, AdaBoost, and XGBoost to tackle the issue of imbalanced datasets. He implemented undersampling, oversampling, ensemble learning, and gradient boosting methods. Afterwards, he trained and tested the optimal model on diverse attack datasets to analyze its performance. Al et al. [20] introduced a novel unsupervised evolutionary IoT zombie network detection method. The main contribution of the proposed method was the utilization of an efficient population intelligence algorithm called the Grey Wolf Optimization algorithm to detect IoT zombie network attacks initiated from compromised IoT devices. It also optimized the hyperparameters of the one-class support vector machine and identified the features that best described the IoT zombie network problem. Salam et al. [21] proposed an improved method for detecting IoT zombie networks by using particle swarm optimization to adjust the hyperparameters of ONE-SVM. This method outperformed existing algorithms in terms of false positive rate, true positive rate, and G-mean value across all categories of IoT devices. Alharbi et al. [22] introduced a graph-based ML model for zombie network detection. The model first considered the importance of graph features and then developed a generalized model for detecting zombie networks based on the selected important features. Maudoux et al. [23] combined preprocessed decision trees and highlighted different types of zombie networks by detecting their intrinsic exchanges. This aggregated IP flows into traffic flows to extract key features and avoid overfitting. Krishnan et al. [24] developed an adaptive voting classifier that could detect all attack categories in the BoT-IoT dataset with higher performance classifier. Shen et al. [25] integrated multiple deep learning models based on a Stacking ensemble learning technique and provided different input feature sets for different primary classifiers to obtain an online detection model for zombie networks. The experiments demonstrated that the proposed ensemble-learning-based online detection method for zombie networks could effectively detect zombie network traffic at multiple stages, with a malicious traffic detection rate of up to 96.47%. Maurya et al. [26] utilized IoT zombie network traffic to design an experimental test platform for generating real-time datasets. Extensive comparative studies were conducted on the proposed dataset and existing datasets using popular machine learning techniques. Waqas et al. [27] evaluated common machine learning algorithms based on performance metrics, and the results showed that tree-based algorithms achieved the highest accuracy in Botnet attack detection on the same sensor devices. Wang et al. [28] introduced zombie network detection architecture based on artificial neural networks, which could better differentiate DGA domains and correctly handle more domains. Gong et al. [29] enhanced the LGBM model by utilizing a Bayesian hyperparameter optimization algorithm and employed it for zombie network detection. Compared to existing models, this model outperformed existing methods in terms of accuracy, false alarm rate, and recall rate. Deng et al. [30] proposed a graph neural-network-based model for detecting IoT zombie networks, which was more easily applicable to unseen data during training compared to other methods.

Currently, many detection models are based on intelligent optimization algorithms. Table 1 summarizes the related research on intelligent optimization algorithms in IoT botnet detection models.
Table 1. Related studies on intelligent optimization algorithms for IoT botnet detection.

<table>
<thead>
<tr>
<th>Year</th>
<th>Reference</th>
<th>Methodology</th>
<th>Dataset</th>
<th>Model</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>Injadat et al. [18]</td>
<td>BO-GP</td>
<td>Bot-IoT</td>
<td>BO-GP-DT</td>
<td>Accuracy: 99.99%</td>
</tr>
<tr>
<td>2020</td>
<td>Al et al. [20]</td>
<td>GWO</td>
<td>N-BaIoT</td>
<td>GWO-OCSVM</td>
<td>G-Mean: 96.8%</td>
</tr>
<tr>
<td>2021</td>
<td>Salam et al. [21]</td>
<td>PSO</td>
<td>NN-BaIoT</td>
<td>PSO-ONE-SVM</td>
<td>G-Mean: 98.7%</td>
</tr>
<tr>
<td>2022</td>
<td>Gong et al. [29]</td>
<td>BO</td>
<td>CTU-13</td>
<td>BO-LGBM</td>
<td>Accuracy: 96.62%, F1 Score: 86.71%</td>
</tr>
</tbody>
</table>

In conclusion, the academic community has made significant progress in research on detecting IoT zombie network attacks. However, due to the complex and ever-changing nature of the IoT environment, as well as the rapid evolution of zombie networks, there is a need for further improvement and refinement in the accuracy of detection models. Additionally, most studies have only conducted experiments on a single dataset, which fails to demonstrate the generalizability of the proposed methods. This work aims to optimize the process of model hyperparameter selection using a novel swarm intelligence optimization algorithm. It is the first time that the DBO algorithm has been combined with the Catboost model and applied in the field of IoT botnet detection, resulting in the achievement of optimal classification results.

3. Methods

3.1. Catboost Algorithm

The Catboost algorithm [31] is a machine learning framework based on the gradient boosting decision tree algorithm. It exhibits excellent classification capabilities when dealing with high-dimensional sparse data. It features adaptive learning rate adjustment, randomized data ordering, automatic handling of missing values and categorical features, and fast training speed. The algorithm also provides numerous hyperparameters and regularization options, facilitating customization and optimization for different application scenarios. Furthermore, the method of using ranking boosts addresses noise points in the training set, avoiding errors caused by gradient estimation and mitigating prediction bias. It effectively handles noisy data in large datasets, such as those found in zombie networks.

3.2. Dung Beetle Optimizer

The dung beetle optimizer (DBO) [32] is a swarm intelligence optimization algorithm inspired by the rolling, foraging, stealing, and breeding behaviors of dung beetles. This algorithm performs both global exploration and local exploitation simultaneously, exhibiting fast convergence. In comparison to other swarm intelligence optimization algorithms, the Dung Beetle Optimizer offers the following advantages:

1. It introduces a novel rolling mechanism for dung beetles, where different search modes utilize information from different time periods to deeply explore the search space and achieve stronger search capabilities, thus avoiding becoming trapped in local optima.
2. The dynamic nature of the parameter “R” stimulates exploration and exploitation within the Dung Beetle Optimizer.
3. Different region-based search strategies, including the spawning region and the best foraging region, promote the developmental behaviors of the Dung Beetle Optimizer, resulting in enhanced global search capabilities.
4. Various update rules ensure a proper balance between local and global search capabilities, maintaining an appropriate equilibrium within the developed Dung Beetle Optimizer.

Overall, the Dung Beetle Optimizer leverages the behavior of dung beetles to offer an efficient and effective optimization algorithm capable of balancing exploration and exploitation for solving complex problems.

3.3. Rules for Updating the Position of Various Dung Beetle Species

The position updating rules for different types of beetles are as follows:
To simulate the rolling behavior of the rolling beetles, they need to move along a given direction within the entire search space. During the rolling process, the position of a rolling beetle is updated and can be represented as follows:

\[x_i(t + 1) = x_i + \alpha \times k \times x_i(t - 1) + b \times \Delta x\]  \hspace{1cm} (1)

\[\Delta x = |x_i(t) - X^w|\]  \hspace{1cm} (2)

where \(t\) represents the current iteration number of the algorithm and \(x_i(t)\) represents the position information of the \(i\)-th beetle at the \(t\)-th iteration. \(k\) is a constant representing the deviation coefficient, \(b\) is a constant belonging to the interval \((0, 1)\), \(\alpha\) is the natural coefficient, and \(\Delta x = |x_i(t) - X^w|\) represents the simulated change in light intensity. \(X^w\) represents the current global worst position.

Additionally, when the beetle’s path encounters obstacles, it needs to change its forward direction through a “dance” behavior. This behavior is mimicked using a tangent function. Once the new path is determined, the beetle’s position is updated using the following formula:

\[x_i(t + 1) = x_i(t) + \tan(\theta)|x_i(t) - x_i(t - 1)|\]  \hspace{1cm} (3)

where \(\theta\) ranges from \([0, \pi]\).

The position updating of breeding beetles requires setting a boundary to simulate the oviposition area of female beetles. It can be defined as:

\[Lb^* = \max(X^* \times (1 - R), Lb)\]  \hspace{1cm} (4)

\[Ub^* = \max(X^* \times (1 + R), Ub)\]  \hspace{1cm} (5)

where \(X^*\) represents the current local best position and \(Ub^*\) and \(Lb^*\) represent the upper and lower boundaries of the oviposition area, respectively. \(R = 1 - t/T_{\text{max}}\) and \(T_{\text{max}}\) represent the maximum number of iterations, while \(Ub\) and \(Lb\) represent the upper and lower bounds of the optimization problem.

After the hatchlings emerge, it is necessary to establish an optimal foraging zone to guide the beetles in their search for food. The optimal foraging zone can be represented as follows:

\[Lb^b = \max(X^b \times (1 - R), Lb)\]  \hspace{1cm} (6)

\[Ub^b = \min(X^b \times (1 + R), Ub)\]  \hspace{1cm} (7)

where \(X^b\) represents the global best position, and other parameters are defined as mentioned above. The position of the foraging beetles is defined as:

\[x_i(t + 1) = x_i(t) + C_1 \times (x_i(t) - Lb^b) + C_2 \times (x_i(t) - Ub^b)\]  \hspace{1cm} (8)

where \(C_1\) represents a random number following a normal distribution, and \(C_2\) represents a random variable belonging to the interval \((0, 1)\).

Finally, concerning the beetles referred to as thieves, they will steal dung balls from other beetles. Since \(X^b\) represents the global best position, it can be assumed that the vicinity of \(X^b\) is the optimal location for food acquisition. During the iteration process, the position updating of the thief beetles can be defined as:

\[x_i(t + 1) = X^b + S \times g \times \left(|x_i(t) - X^*| + |x_i(t) - X^b|\right)\]  \hspace{1cm} (9)

where \(g\) is a random vector of size \(1 \times D\) following a normal distribution, and \(S\) represents a constant.
3.4. Improved Dung Beetle Optimizer

Controlling the quality of the initial population in DBO is crucial for enhancing the algorithm performance. The quality is primarily determined by two factors: the search range and the initial positions. If the search range is too narrow, it hampers the algorithm’s ability to explore. Conversely, if the initial positions are close to the global optimal solution, the population can effectively uncover valuable information within a superior solution space. However, the standard DBO adopts a random initialization method to generate the population, which makes it difficult to meet the above two requirements.

To address this issue, this paper introduces the centroid opposition-based learning strategy (COBL) [33] to generate the initial population. The basic idea is to calculate the reverse solution based on the centroid of the population. This approach allows for searching high-quality solutions in a wider range, effectively improving the quality of the initial population. The initialization of the population using the centroid opposition-based learning strategy is as follows:

Let \((X_1, \ldots, X_n)\) be \(n\) points in a D-dimensional search space with unit weight. The centroid of the entire set is defined as follows:

\[
M = \frac{X_1 + \cdots + X_n}{n}
\]

then we have:

\[
M_d = \frac{1}{n} \sum_{i=1}^{n} x_{d,i}, i = 1, 2, \ldots, n, \ d = 1, 2, \ldots, D
\]

If the centroid of a discrete and uniformly distributed set is denoted \(M_d\), then the opposite-point \(\hat{X}_i\) of a point is defined as follows:

\[
\hat{X}_i = \begin{cases} 
    x_{d}^{\text{min}} + r \times (M_d - x_{d}^{\text{min}}), & \hat{X}_i < x_{d}^{\text{min}} \\
    M_d + r \times (x_{d}^{\text{max}} - M_d), & \hat{X}_i > x_{d}^{\text{max}}
\end{cases}
\]

In the equation, \(x_{d}^{\text{min}}\) and \(x_{d}^{\text{max}}\) represent the upper and lower limit values in the D-dimensional space, respectively. \(r\) is a uniformly distributed random number on \([0, 1]\).

3.5. IDBO-Catboost Model

Optimizing the hyperparameters of a machine learning model can significantly impact its performance [34]. However, tuning hyperparameters can be time-consuming and computationally expensive. Hyperparameter tuning can be carried out through manual trial-and-error, or by using relatively simple algorithms such as random search, grid search, or Bayesian optimization. While manual tuning is straightforward, it often leads to local optima. On the other hand, random search and grid search can partially mitigate these issues, but are less efficient when dealing with high-dimensional parameter spaces. Bayesian optimization offers a more efficient approach for finding optimal hyperparameters, but it requires defining a probability model and may demand additional computational resources to train the model.

In comparison to these methods, the dung beetle optimization algorithm is a novel swarm intelligence optimization algorithm that demonstrates fast convergence and strong global search capabilities. It can effectively discover the optimal combination of parameters in machine learning models.

When predicting botnet attacks using the Catboost model, inappropriate parameter settings can have a significant impact on the model’s predictions. Therefore, in this paper an improved dung beetle optimization (IDBO) algorithm was chosen to optimize the important hyperparameters of the Catboost model. The ensemble model solves the problem of the
Catboost model easily, becoming stuck in local optima while further improving the accuracy of the predictions.

The design concept of the IDBO-Catboost detection model combines the dung beetle optimizer algorithm with the Catboost algorithm, optimizing the key parameters in Catboost that have the most significant impact on classification performance. This approach leverages the parameter optimization capabilities of the DBO algorithm and combines it with the strengths of Catboost in handling classification problems, achieving an organic integration of the two. It eliminates the tedious manual tuning process. The IDBO-Catboost detection model is illustrated in Figure 1.

**Figure 1.** IDBO-Catboost Botnet Detection Model.

The optimization process of Catboost using the improved dung beetle optimizer can be described in the following steps:

Step 1: Initialize the Catboost model parameters and determine the optimal range based on experience. Set algorithm control parameters such as population size, maximum number of iterations, and position boundaries. Randomly define the positions of the beetles in the search space and form a new population according to the centroid opposition-based learning strategy.

Step 2: Calculate the fitness values of all beetles based on the objective function.

Step 3: Start the loop and update the positions of the beetles.

Step 4: Check if each beetle is out of bounds.

Step 5: Calculate the individual best and global best solutions, as well as their fitness values, based on the current positions of the beetles. The individual best represents the best solution found by each beetle, and the global best is selected from these individual best solutions.
Step 6: Repeat the above steps until the stopping criteria are met—either the maximum number of iterations is reached or the model performance reaches a pre-defined threshold. When the algorithm finishes, the global best solution and its fitness value are outputted. The global best solution represents the optimal parameter combination for the Catboost algorithm, and the fitness value represents the classification performance of the model in terms of AUC (Area Under the Curve).

In this experiment, the parameters that have the most significant impact on the performance of the Catboost algorithm are the number of gradient boosting trees (iterations), learning_rate, and maximum tree depth (depth). The dung beetle optimizer is used to search for the optimal parameters for these parameters. The pseudocode for constructing the IDBO-Catboost model is shown in Algorithm 1.

**Algorithm 1: IDBO-Catboost Pseudocode**

**Input:** A Catboost model with n hyperparameters, a training set, the maximum number of iterations ($T_{\text{max}}$) for the dung beetle optimizer, a population size (N) for the dung beetle optimizer, and upper bound (Ub) and lower bound (Lb) for the dung beetle optimizer

**Output:** IDBO-Catboost Model

1. Initialization: population size, search dimensionality, maximum number of iterations, search range.
2. The centroid opposition-based learning strategy forms a new population.
3. **While** iterations $t < T_{\text{max}}$ **do**
   4. for $i \leftarrow 1$ to N **do**
   5. if $i ==$ rolling beetle then
   6. $\delta = \text{rand}(1)$
   7. if $\delta < 0.9$ then
   8. Update the position of the rolling beetle using Equation (1)
   9. else
   10. Update the position of the rolling beetle using Equation (3)
   11. end if
   12. end if
   13. if $i ==$ breeding beetle then
   14. Update the position of the breeding beetle using Equations (4) and (5)
   15. end if
   16. if $i ==$ foraging beetle then
   17. Update the position of the foraging beetle using Equation (8)
   18. end if
   19. if $i ==$ thief beetle then
   20. Update the position of the thief beetle using Equation (9)
   21. end if
   22. end for
   23. if the newly generated position is better than the previous position then
   24. Update the corresponding position
   25. end if
   26. Iteration $t = t + 1$
   27. end while
28. return the optimal position (best hyperparameter combination for Catboost) and the fitness value (AUC value).
29. Build the IDBO-Catboost model

4. Experimental Results

In this section, the performance of the proposed improved DBO-Catboost detection model is compared and evaluated using publicly available datasets.
4.1. Experimental Environment

The hardware platform used in this experiment included CPU: 12th Generation Intel Core i5 12400f, GPU: NVIDIA GeForce 3060 Ti, 32 GB 3200 Mhz RAM, and a 1 TB SSD for storage.

The software platform for this experiment was conducted on Windows 10 operating system, using Python 3.7 as the primary programming language. Various tools and packages such as pandas, sklearn, and numpy were employed for the experiment.

4.2. Dataset Introduction

The dataset used in this experiment was the Botnet dataset [35], created by the Canadian Institute for Cybersecurity. The dataset was divided into training and testing sets. The details of the dataset are presented in Table 2.

Table 2. Basic Information of Botnet Dataset.

<table>
<thead>
<tr>
<th></th>
<th>Number of Samples</th>
<th>Number of Features</th>
<th>Attack</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>331,852</td>
<td>84</td>
<td>173,192</td>
<td>158,660</td>
</tr>
<tr>
<td>Testing Set</td>
<td>345,746</td>
<td>84</td>
<td>212,154</td>
<td>133,592</td>
</tr>
</tbody>
</table>

4.3. Data Preprocessing

Data preprocessing is an essential step in machine learning, involving cleaning, transforming, and standardizing raw data before feeding it into a model. The main purpose of data preprocessing is to improve data quality, enhance model performance, and increase generalization ability.

Typically, data preprocessing involves several steps:

Step1: Data cleaning. The primary objective is to remove duplicate and invalid values from the dataset. This also involves handling missing values by imputation.

Step2: Data type conversion. This refers to the process of converting the data type of features to a different format. Machine learning algorithms typically require numerical data, so non-numeric features need to be converted into a numerical format for the algorithms to process.

Step3: Standardization. Some features in the dataset may have varying scales or ranges of values. To simplify the learning process of the model, it is necessary to standardize the features in the dataset.

Step4: Feature engineering. The Botnet dataset consists of 83 features, many of which are redundant. These redundant features can negatively impact the model training. In this study, principal component analysis (PCA) [36] is used for feature dimensionality reduction.

It is worth noting that Catboost has strong capabilities in handling missing values. It can automatically detect and handle missing data, eliminating the need for additional feature engineering and reducing the time required for data preprocessing before training. In this section, to ensure consistency across all experiments, the preprocessed dataset was used.

4.4. Evaluation Metrics

This article utilized evaluation metrics such as the confusion matrix, precision, recall, accuracy, and F1 score to analyze and assess the classification results. The confusion matrix is defined as shown in Table 3.

Table 3. Confusion Matrix.

<table>
<thead>
<tr>
<th></th>
<th>Predicted Positive</th>
<th>Predicted Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Positive</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Actual Negative</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>
Accuracy: refers to the proportion of correctly predicted samples out of the total samples:
\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (14)
\]

Precision: refers to the probability of correctly predicting a positive instance:
\[
\text{Precision} = \frac{TP}{TP + FP} \times 100\% \quad (15)
\]

Recall: refers to the probability of correctly predicting a positive instance among all actual positive samples:
\[
\text{Recall} = \frac{TP}{TP + FN} \times 100\% \quad (16)
\]

F1 Score: represents the harmonic mean of precision and recall:
\[
\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100\% \quad (17)
\]

4.5. Experimental Results

The Catboost algorithm had a significant impact on the classification performance through parameters such as the maximum number of trees (iterations), tree depth (depth), and learning rate (learning_rate). In this study, the dung beetle optimizer was used to optimize these parameters for Catboost. When using DBO-Catboost, the iteration count was set to 100. After several preliminary experiments, it was observed that 30 iterations were sufficient for the DBO-Catboost model to converge. Therefore, the iteration count was set to 30, the number of beetle populations was set to 40, and the parameter search ranges were set as follows: Iterations $\in [500, 1000]$, depth $\in [4, 16]$, learning_rate $\in [0.01, 0.2]$.

To verify the optimization effectiveness of the improved DBO algorithm on the model, particle swarm optimization (PSO), grey wolf optimization (GWO), and the original dung beetle optimization (DBO) algorithms were used for comparison. The number of iterations was set to 30, and the AUC value of the model was used as the fitness function. The results are shown in Figure 2.

![Figure 2. Cont.](image-url)
Figure 2. Changes in AUC values for the number of iterations. (a) IDBO-Catboost, (b) DBO-Catboost, (c) PSO-Catboost, and (d) GWO-Catboost.

From the results in Figure 2, it can be observed that the improved DBO algorithm achieved faster convergence, reaching convergence after 16 iterations. In terms of the AUC value, after 30 iterations the IDBO-Catboost model had the highest AUC value, indicating better optimization performance. Comparing the improved DBO algorithm with the unimproved DBO algorithm, the improved DBO algorithm had a higher initial fitness value, and the final fitness value was also higher, with a faster iteration speed. This indicates that the improved DBO generated a better initial population that was closer to the optimal solution, requiring fewer iterations to converge on the vicinity of the best solution. This result validates the effectiveness of our improvement in the generation rules of the initial DBO population.

The experimental results on the Botnet dataset, comparing Catboost, PSO-Catboost, GWO-Catboost, and the original DBO-Catboost models, are shown in Table 4.

Table 4. Comparison of experimental results.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catboost</td>
<td>94.09%</td>
<td>93.93%</td>
<td>93.93%</td>
<td>93.93%</td>
</tr>
<tr>
<td>DBO-Catboost</td>
<td>95.84%</td>
<td>95.75%</td>
<td>95.75%</td>
<td>95.75%</td>
</tr>
<tr>
<td>IDBO-Catboost</td>
<td>96.21%</td>
<td>96.10%</td>
<td>96.10%</td>
<td>96.10%</td>
</tr>
<tr>
<td>PSO-Catboost</td>
<td>95.00%</td>
<td>94.89%</td>
<td>94.89%</td>
<td>94.89%</td>
</tr>
<tr>
<td>GWO-Catboost</td>
<td>94.83%</td>
<td>94.72%</td>
<td>94.72%</td>
<td>94.72%</td>
</tr>
</tbody>
</table>

According to the information in Table 4, the optimization strategy of IDBO-Catboost effectively found the best parameter combination for the model, improving its performance. In a vertical comparison with Catboost, IDBO-Catboost outperformed Catboost with an improvement of 2.17% in accuracy and F1 score. In a horizontal comparison, DBO-Catboost outperformed PSO-Catboost and GWO-Catboost with improvements of 1.21% and 1.38% in F1 score and accuracy, respectively. As for the comparison with the original DBO-Catboost, the IDBO-Catboost showed a slight improvement of 0.35% in accuracy and F1 score. Although the improvement was modest, it still represents a significant value when applied in real-life scenarios. The main reason for these differences is that the improved DBO algorithm can increase the population size and diversity of the population by optimizing the oviposition area, thereby better utilizing the search space. Additionally, by optimizing the strategy for generating the initial population, the algorithm can come closer to the optimal solution within the search space, enhancing its search capability. Therefore, the improved DBO algorithm can comprehensively explore the search space and achieve better global search capability to obtain the optimal parameter combination for the Catboost algorithm.

Figure 3 shows the ROC curves of several models. According to the information in Figure 3, we can see that the ROC curve of IDBO-Catboost was closer to the top-left corner,
indicating that IDBO-Catboost achieved a better balance between sensitivity and specificity. Additionally, the AUC value of IDBO-Catboost was the highest among all the comparison models, reaching 0.99. Therefore, this result effectively demonstrates that the model has good performance and high discriminative ability at different thresholds.

![Receiver Operating Characteristic (ROC) curves](image)

Figure 3. AUC values and ROC curves of different models.

After validating the optimization effect of IDBO on the Catboost algorithm, we compared it with the recently proposed models, namely BO-GP-DT (2020), GWO-OCSVM (2020), PSO-ONE-SVM (2021), BO-LGBM (2022), and the original DBO-Catboost. The parameter settings for the models are shown in Table 5, and the experimental results from the Botnet dataset are presented in Table 6.

Table 5. Parameter settings.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Parameter Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>BO-GP-DT</td>
<td>max_depth = 6, max_leaf_nodes = 11, min_samples_leaf = 2, min_samples_split = 7</td>
</tr>
<tr>
<td>BO-LGBM</td>
<td>max_depth = 18, learning_rate = 0.58, n_estimators = 800, num_leaves = 35</td>
</tr>
<tr>
<td>DBO-Catboost</td>
<td>Iterations = 849, learning_rate = 0.12, depth = 8</td>
</tr>
<tr>
<td>IDBO-Catboost</td>
<td>Iterations = 816, learning_rate = 0.17, depth = 7</td>
</tr>
<tr>
<td>GWO-OCSVM</td>
<td>( \nu = 0.061, Y = 0.21, \text{Kenel} = \text{rbf} )</td>
</tr>
<tr>
<td>PSO-ONE-SVM</td>
<td>( \nu = 0.053, Y = 0.22, \text{Kenel} = \text{rbf} )</td>
</tr>
</tbody>
</table>

Table 6. Experimental Comparison of Various Models on Botnet.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BO-GP-DT</td>
<td>95.19%</td>
<td>95.06%</td>
<td>95.06%</td>
<td>95.06%</td>
</tr>
<tr>
<td>BO-LGBM</td>
<td>95.53%</td>
<td>95.47%</td>
<td>95.47%</td>
<td>95.47%</td>
</tr>
<tr>
<td>DBO-Catboost</td>
<td>95.84%</td>
<td>95.75%</td>
<td>95.75%</td>
<td>95.75%</td>
</tr>
<tr>
<td>IDBO-Catboost</td>
<td>96.21%</td>
<td>96.10%</td>
<td>96.10%</td>
<td>96.10%</td>
</tr>
<tr>
<td>GWO-OCSVM</td>
<td>95.45%</td>
<td>95.38%</td>
<td>95.38%</td>
<td>95.38%</td>
</tr>
<tr>
<td>PSO-ONE-SVM</td>
<td>95.62%</td>
<td>95.55%</td>
<td>95.55%</td>
<td>95.55%</td>
</tr>
</tbody>
</table>
From the results in Table 6, it can be observed that IDBO-Catboost achieved the highest accuracy of 96.21% among all the comparative models. It outperformed BO-GP-DT, BO-LGBM, GWO-OCSVM, PSO-ONE-SVM, and the original DBO-Catboost by 1.04%, 0.63%, 0.72%, 0.55%, and 0.35%, respectively. Similarly, IDBO-Catboost also demonstrated superior precision, recall, and F1 score compared to other models, indicating its optimal classification performance.

There are two main reasons for these results: firstly, compared to decision trees and SVM algorithms, gradient-boosting tree-based algorithms such as Catboost and LGBM perform better when dealing with large-scale datasets, and the Catboost algorithm is particularly effective in handling noisy data in the dataset. Secondly, for optimization algorithms, the ability to balance global and local searches is crucial. In the dung beetle optimizer, the breeding behavior ensures that new individuals have better fitness, foraging behavior accelerates the speed of local search, stealing behavior utilizes better solutions discovered in the global search, and rolling behavior increases the algorithm’s diversity, helping it to escape local optima and further improving the global search capability. Finally, the improved DBO algorithm generates an initial population that is closer to the optimal solution, allowing it to better find the optimal parameter combination. Through the combined effect of these behaviors, the algorithm achieves a balance between global and local search, thereby enhancing its search ability and optimization performance, effectively improving the detection and classification performance of the model.

To evaluate the generalization ability of the proposed method, the Bot-IoT dataset [37] was used for performance assessment. Due to the severe class imbalance in the Bot-IoT dataset, where there was a small amount of normal data and a large amount of attack data, the dataset was processed using a combination of the SMOTE technique and an undersampling technique to achieve a balanced dataset. The data preprocessing workflow is shown in Figure 4, and the sample distribution before and after dataset processing is illustrated in Figure 5.

![Figure 4. Imbalanced data handling process.](image)

![Figure 5. Dataset sample distribution.](image)
Table 7 presents the performance of the IDBO-Catboost and the comparison models on the processed Bot-IoT dataset.

Table 7. Experimental Comparison of Various Models on BotIoT.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catboost</td>
<td>94.38%</td>
<td>94.25%</td>
<td>94.25%</td>
<td>94.25%</td>
</tr>
<tr>
<td>BO-GP-DT</td>
<td>96.73%</td>
<td>96.68%</td>
<td>96.68%</td>
<td>96.68%</td>
</tr>
<tr>
<td>BO-LGBM</td>
<td>97.10%</td>
<td>97.06%</td>
<td>97.06%</td>
<td>97.06%</td>
</tr>
<tr>
<td>DBO-Catboost</td>
<td>98.13%</td>
<td>98.12%</td>
<td>98.12%</td>
<td>98.12%</td>
</tr>
<tr>
<td>IDBO-Catboost</td>
<td>98.62%</td>
<td>98.57%</td>
<td>98.57%</td>
<td>98.57%</td>
</tr>
<tr>
<td>GWO-OCSVM</td>
<td>96.54%</td>
<td>96.50%</td>
<td>96.50%</td>
<td>96.50%</td>
</tr>
<tr>
<td>PSO-ONE-SVM</td>
<td>97.08%</td>
<td>97.01%</td>
<td>97.01%</td>
<td>97.01%</td>
</tr>
</tbody>
</table>

From the information in Table 7, it can be observed that compared to other models, IDBO-Catboost achieved higher detection accuracy on the Bot-IoT dataset, with an accuracy and F1 score of 98.57%. This indicates that the hyperparameters identified by IDBO-Catboost were highly effective, and the model exhibited good generalization ability. The main reason for this result was the presence of the dynamically changing parameter R in the dung beetle optimizer, which allowed the algorithm to adapt better to different problems and enhance the model’s generalization performance.

In addition to performance metrics such as accuracy and F1 score, the average time (sum of model training and prediction time) of the proposed model and other detection models on the Botnet dataset and Bot-IoT dataset are compared in Table 8. It can be observed that the IDBO-Catboost detection model requires less time compared to the other detection models. This is because the IDBO-Catboost model can utilize GPU acceleration for training, thereby improving the detection efficiency of the model. Regarding DBO-Catboost and Catboost, the IDBO-Catboost model has fewer iterations, resulting in a shorter training time compared to DBO-Catboost and Catboost models.

Table 8. Average time.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDBO-Catboost</td>
</tr>
<tr>
<td>Botnet</td>
<td>38.21</td>
</tr>
<tr>
<td>BotIoT</td>
<td>20.19</td>
</tr>
</tbody>
</table>

In summary, this paper conducted experiments using two datasets, Botnet and Bot-IoT, related to Internet of Things (IoT) botnet networks. By comparing with existing detection models, the proposed improved DBO-Catboost detection model demonstrated the best classification performance in terms of high accuracy and detection efficiency. Therefore, it can be effectively applied to the detection tasks of IoT botnet networks.

5. Conclusions

To address the issue of designing an effective zombie network detection model and improving detection performance, this paper proposes a zombie network detection model based on the improved DBO-CatBoost. We improved the initial population generation strategy of the dung beetle optimization algorithm, replacing random initialization with the centroid opposition-based learning strategy, and applied it to the field of IoT botnet network detection. The improved DBO-CatBoost model utilized the dung beetle optimizer to optimize the hyperparameters of CatBoost, reducing the cumbersome and repetitive process of hyperparameter tuning and improving efficiency. The proposed method was validated using real-world IoT traffic datasets. Experimental results demonstrate that the proposed approach achieved accuracy rates of 96.10% and 98.57% on the Botnet and Bot-IoT...
datasets, respectively. It outperformed all the compared models in terms of classification performance and exhibited a certain level of generalization capability.

There are some limitations in this study that need to be addressed. When dealing with imbalanced datasets, further research is needed to improve the focus on the minority class and enhance the detection performance of the model on imbalanced data. Additionally, in the future, introducing interpretability mechanisms will be beneficial to improve the explainability of the model.

**Author Contributions:** Conceptualization, C.Y.; Writing—original draft, C.Y.; experiment, C.Y.; resources, C.Y.; formal analysis, C.Y. and W.G.; project running, W.G. and Z.F.; supervision, W.G. and Z.F.; funding acquisition, Z.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China (grant no. 11864005).

**Data Availability Statement:** The dataset used can be located at https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 20 April 2023) and https://www.unb.ca/cic/datasets/botnet.html (accessed on 22 April 2023).

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

**References**


