Hierarchical Classification of Botnet Using Lightweight CNN

Worku Gachen Negera 1, Friedhelm Schwenker 2, Degaga Wolde Feyisa 3, Taye Girma Debelee 3,4,* and Henock Mulugeta Melaku 1

1 Addis Ababa Institute of Technology, Addis Ababa University, 445, Ethiopia; worku.gachena2@gmail.com (W.G.N.); henock.mulugeta@aait.edu.et (H.M.M.)
2 Institute of Neural Information, University of Ulm, 89069 Ulm, Germany; friedhelm.schwenker@uni-ulm.de
3 Ethiopian Artificial Intelligence Institute, Addis Ababa 40782, Ethiopia; degagawolde@gmail.com
4 Department of Electrical and Computer Engineering, Addis Ababa Science and Technology University, Addis Ababa 16417, Ethiopia

* Correspondence: tayegirma@gmail.com

Abstract: This paper addresses the persistent threat of botnet attacks on IoT devices, emphasizing their continued existence despite various conventional and deep learning methodologies developed for intrusion detection. Utilizing the Bot-IoT dataset, we propose a hierarchical CNN (HCNN) approach featuring three levels of classification. The HCNN approach, presented in this paper, consists of two networks: the non-hierarchical and the hierarchical network. The hierarchical network works by combining features obtained at a higher level with those of its descender. This combined information is subsequently fed into the following level to extract features for the descendant nodes. The overall network consists of 1790 parameters, with the hierarchical network introducing an additional 942 parameters to the existing backbone. The classification levels comprise a binary classification of normal vs attack in the first level, followed by 5 classes in the second level, and 11 classes in the third level. To assess the effectiveness of our proposed approach, we evaluate performance metrics such as Precision (P), Recall (R), F1 Score (F1), and Accuracy (Acc). Rigorous experiments are conducted to compare the performance of both the hierarchical and non-hierarchical models and existing state-of-the-art approaches, providing valuable insights into the efficiency of our proposed hierarchical CNN approach for addressing botnet attacks on IoT devices.

Keywords: botnet; botnet attack; Bot-IoT; classification; CNN; hierarchical; lightweight

1. Introduction

The Internet of Things (IoT) is a dynamic ecosystem growing quickly in scale, connectivity, and variety of applications. This ubiquitous ecosystem is similar to other emerging technologies in that it permeates every aspect of our existence. Unfortunately, despite all the benefits that the Internet of Things brings, the increased attack surface that it creates has become even more significant. Vulnerabilities are made worse by the intrinsic limitations of devices, which frequently lack strong security mechanisms [1]. There has been a noticeable trend recently in which the IoT ecosystem is becoming more compromised by botnet threats [2]. An enormous network of infected IoT devices can have devastating consequences since it can grow quickly and launch powerful attacks. As a result, finding practical ways to strengthen IoT systems has emerged as a crucial and complex field of study. Among these, machine learning-based methods are particularly attractive since they allow for the early identification of possible assaults and the identification of unusual behavior [1].

In the realm of cybersecurity, an IoT botnet integrates itself into a network by executing a series of malicious operations on computing devices. These operations comprise three key steps: first, identifying susceptible devices through scanning; next, installing a compatible bot tailored to the vulnerable device’s architecture for propagation; and finally, launching an
attack through a command and control operation, as elucidated by Wazzan [3]. A concrete example is the Mirai botnet, which encompasses attack vectors, a scanning process actively seeking other devices for compromise, and a command and control system governing the compromised devices (bots). This orchestrates further propagation and instigates attacks, exemplifying the sophisticated nature of contemporary botnet strategies [4].

Contribution

The paper’s contribution is the creation and application of a novel method for the hierarchical classification of botnet attacks using Convolutional Neural Networks (CNN) on the Bot-IoT datasets. The persistent threat of botnet attacks on Internet of Things (IoT) devices in the field of cybersecurity necessitates the use of strong and efficient detection systems. The hierarchical CNN architecture presented in this paper improves the precision and level of detail in identifying botnet attacks. The contribution of the paper is summarized as follows:

- Our suggested method aims to enhance the efficiency of botnet attack classification on the Bot-IoT dataset by leveraging its inherent hierarchical structure.
- Additionally, the proposed CNN model is lightweight, demanding less memory and execution time, rendering it suitable for deployment on compact IoT devices.
- This approach is crucial in addressing the evolving IoT security landscape, as it introduces advanced hierarchical categorization algorithms. These algorithms facilitate more precise and nuanced identification of botnet activity

The rest of the paper is organized as follows. The related works are presented in Section 2. In Section 3, the dataset description, proposed approach, and evaluation metrics are covered. In Section 4, the experimental results and discussion followed by a comparison to the existing approach are discussed. Finally, Section 5 presents the conclusion.

2. Related Work

Botnet attack vulnerabilities still exist in Internet of Things (IoT) networks, even with the introduction of SDN (Software-Defined Networking) [5,6]. There are several ways to classify botnet attacks, and particularly in SDN-enabled IoT networks, bots use network-probing attacks and backdoor vulnerabilities to obtain a foothold. A variety of attacks, such as DDoS (Distributed Denial of Service), DoS (Denial of Service), scanning assaults, and information theft, can be carried out using this vulnerable position [7–9]. Conventional neural network-based Network Intrusion Detection Systems (NIDS) tend to incur high resource consumption, making them impractical for deployment in Internet gateways, routers, and Internet of Things (IoT) devices [10,11].

Wei et al. [10] introduced a lightweight, two-stage NIDS for IoT networks, exclusively utilizing packet-length features. The system efficiently detects botnet activities in resource-limited devices. In their approach, they involved 21 discriminative statistical features for distinguishing between malicious and normal traffic flows. In the first stage, an autoencoder-based module filters out a significant portion of normal traffic. Subsequently, a novel mechanism transforms packet length sequences into RGB images for malicious traffic classification using a lightweight CNN. The authors in [12] proposed a feature selection using Information Gain (IG) and Gain Ratio (GR) with the ranked top 50% features for the detection of DoS and DDoS attacks. The proposed approach achieved 99.9993% accuracy with 16 selected features.

Hybridized deep learning-driven botnet malware detection algorithms, as explored by Liaqat [13], have integrated convolutional neural networks (CNNs) to extract features effectively. While CNN’s convolutional and pooling layers excel at capturing spatial features, the challenge lies in their limited grasp of temporal information, hindering the identification of feature interdependency in a 2D-CNN. In response to this limitation, Liaqat et al. [13] proposed an innovative solution by incorporating cuDNNLSTM layers after the CNN layers, resulting in a hybrid deep learning architecture. The CNN-BiLSTM method proposed in [14] is well-known for its convolutional and bidirectional long short-
term memory architecture. The effectiveness of this hybrid model was demonstrated through training and testing on the Bot-IoT dataset, with reported results showcasing an impressive accuracy of 99.99%, precision of 99.99%, recall of 99.99%, and an F1-score of 99.99%. These findings underscore the superiority of the CNN-BiLSTM architecture in successfully detecting botnet malware within Software-Defined Networking (SDN)-enabled Internet of Things (IoT) networks.

In [15], the novel hierarchical CNN-attention network, CANET, is proposed. Within CANET, local spatiotemporal feature extraction is the primary focus of the CNN-Attention (CA) Block, which is formed by merging CNN with the attention mechanism. More suited for contemporary large-scale network intrusion detection, the multi-layer CA Block combination can fully understand the multi-level spatiotemporal aspects of network attack data. Additionally, they suggest using Equalisation Loss v2 (EQL v2) to balance learning attention on minority classes and enhance the minority class weight to address the issue of class imbalance. Numerous tests show that CANET works better in terms accuracy, detection rate, and false positive rate than the most advanced techniques. Additionally, it effectively raises the minority class detection rate.

Dina et al. [16] introduced the concept of focused loss, a specialized loss function, to tackle the data imbalance problem in IoT intrusion detection. By using dynamically scaled-gradient updates to prioritize difficult negatives and automatically reduce the influence of simple examples, this function helps to train machine learning models that are incredibly successful. The proposed method was evaluated against the most advanced intrusion-detection models through comprehensive experimental evaluations on three datasets covering various IoT domains. The results showed that their method, which trained deep learning models using the focal loss function, performs better than the conventional cross-entropy loss function in terms of accuracy, precision, F1 score, and Matthews Correlation Coefficient (MCC) score.

Xu et al. [17] employed the binary grey wolf optimizer (BGWO) heuristic algorithm and recursive feature elimination (RFE) to select features for their intrusion detection system. They used the synthetic minority oversampling technique (SMOTE) to oversample the minority classes. Finally, they applied XGBoost as a classifier and achieved a perfect F1 score of 1.0 for the five-class classification on the Bot-IoT dataset. Similarly, the study conducted by Alosaimi et al. [18] involved a comparison of decision trees, ensemble bag, K-nearest neighbor, linear discriminant, and support vector machine in the context of 2, 5, and 11 class classification on the dataset. The authors asserted that the ensemble bag exhibited 100% accuracy across all hierarchical levels in the dataset. However, they refrained from disclosing the specific models that were ensembled together and neglected to provide detailed information regarding the computational resource requirements for this approach.

3. Materials and Method

In this study, we will use a step-by-step method, depicted in Algorithm 1, to identify and categorize botnet attacks using HCNN. To comprehend intricate patterns and time-related facets of botnet activity, we put out new lightweight HCNN models. Precision, recall, and F1-score are examples of regularly used assessment criteria that we employed to assess the effectiveness of our suggested model. Our objective is to use cutting-edge deep learning techniques to develop a dependable and useful system for identifying and categorizing botnet attacks.
Algorithm 1 Data Preprocessing and Model Training

1: procedure DATAPREPROCESSING
2:   Downsampling the dataset
3:   Remove columns with empty entries
4:   Drop rows with NaN entries
5:   Convert categorical values to label encoded data
6:   Normalize the data using Standard Scaler
7: end procedure
8: procedure BUILDMODEL
9:   Build a CNN model with a small number of parameters
10: end procedure
11: procedure TRAINMODEL
12:   Split the dataset into training, validation, and test datasets
13:   Define hyperparameters and set up the experiment setup
14:   Train the model using the training and validation dataset
15: end procedure
16: procedure EVALUATEMODEL
17:   Evaluate the model using learning curves
18:   Evaluate the model on the testing dataset
19: end procedure
20: procedure CHECKSATISFACTION
21:   if not satisfied then
22:     Repeat from Data Preprocessing
23:   end if
24: end procedure
25: procedure REPORTRESULTS
26:   if satisfied then
27:     Report the results
28:   end if
29: end procedure

3.1. Dataset

The Bot-IoT dataset, as discussed in [19] serves as a valuable resource for researchers and practitioners involved in network security within the Internet of Things (IoT) domain. The dataset, introduced by the authors, is designed to address the growing need for realistic and representative datasets that can aid in the analysis of botnets operating within IoT networks.

The paper emphasizes the importance of understanding and countering security threats posed by botnets in the dynamic landscape of IoT. The dataset is crafted to capture various aspects of IoT device behavior, network traffic, and communication patterns, encompassing both normal and malicious activities. It facilitates the development, testing, and evaluation of detection and prevention mechanisms specifically tailored to IoT-based botnets. Researchers can leverage the Bot-IoT dataset to conduct experiments, validate algorithms, and advance the field of IoT security. The ultimate goal is to enhance the capabilities of network forensic analytics, providing insights and solutions to combat the evolving challenges associated with botnets in the IoT ecosystem. The dataset’s significance lies in its contribution to creating a more secure IoT environment through the development and refinement of cutting-edge security measures [19].

The dataset consists of 72 million records and encompasses 42 features, rendering it notably extensive with a substantial size of 17.87 GB. The dataset contains a hierarchical structure as depicted in Figure 1. The dataset exhibits imbalance, evident in the significant variation among the instance counts for each class, as illustrated in Figures 2 and 3. Due to the extensive size of the dataset, making it impractical to train the proposed method, we addressed this challenge by undersampling classes with more than one million instances.
Specifically, we randomly selected 20% from each of these classes while retaining all samples for the remaining classes.

**Figure 1.** The hierarchical structure of the Bot-IoT dataset.

**Figure 2.** Number of samples for each class in the second-level hierarchy.

**Figure 3.** Number of samples for each class in the third-level hierarchy.

During the data cleaning phase, columns containing over 20% missing values are eliminated. Subsequently, for the remaining records, any rows with null values are excluded. Categorical columns undergo encoding through label encoding, and the data are then divided into X for input and Y for the target variable. The input data (X) are further
normalized using a standard scaler. The dataset is divided into training, validation, and testing sets using random selection, with a split ratio of 80%, 10%, and 10%, respectively.

3.2. Proposed Approach

Hierarchical classification (HC) has demonstrated remarkable effectiveness in handling datasets with inherent hierarchical structures, according to several studies [20–25]. Their success can be attributed to their ability to capture information at different levels of granularity within the data. HC achieves this by employing a hierarchical architecture that mimics the hierarchical nature of the data itself. This allows them to learn informative features at each level of the hierarchy, ultimately leading to improved classification performance.

Numerous investigations have highlighted the efficiency of employing CNN in the classification of tabular data [10,11,13]. In the present research, we introduce a lightweight CNN model tailored specifically for this purpose. As presented in the Section 3.1, the dataset at hand exhibits a hierarchical arrangement. Consequently, we put forth a novel approach employing a Hierarchical Convolutional Neural Network (HCNN). This architecture utilizes a common backbone depicted in Table 1, yet diverges into distinct models corresponding to each level within the hierarchical structure. This strategy enables a comprehensive exploration of the HCNN’s impact by comparing the performance across different hierarchy levels.

Table 1. Base model architecture [AF = Activation Function].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Layers</th>
<th>Output Shape</th>
<th>Kernel</th>
<th>AF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-GAP Backbone</td>
<td>Conv Layer</td>
<td>(26, 16)</td>
<td>(3, 3)</td>
<td>tanh</td>
</tr>
<tr>
<td></td>
<td>MaxPooling</td>
<td>(13, 16)</td>
<td>(2, 2)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Conv Layer</td>
<td>(13, 16)</td>
<td>(3, 3)</td>
<td>tanh</td>
</tr>
<tr>
<td></td>
<td>GAP Layer</td>
<td>(1, 16)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The hierarchical feature learning process in CNNs allows the network to generalize well to new, unseen data. The learned hierarchical representations can capture essential characteristics of the input data, leading to better generalization performance on diverse datasets. While the initial level of the HCNN might not yield a substantial alteration in classification performance, its impact becomes notably evident in the final level.

This study introduces two models: a non-hierarchical model, illustrated in Figure 4, and a hierarchical model, illustrated in Figure 5. The non-hierarchical models incorporate a backbone as shown in Table 1, along with their respective final dense layer for classification employing a softmax Activation Function. The hierarchical model utilizes the same backbone but incorporates a hierarchical structure at each level of the hierarchy.

The primary structure of the backbone model includes Conv1D, MaxPooling, Conv1D, and Global Average Pooling (GAP). The adoption of Global Average Pooling (GAP) in place of the flattened layer has resulted in a decrease in the overall parameter count. The total parameters in the backbone model sum up to 848, reflecting a notably compact size. For the three non-hierarchical models, the parameter counts are 882 for the first model Figure 4a, 933 for the second model Figure 4b, and 1035 for the final model Figure 4c.
Figure 4. Developing distinct models for each hierarchical level results in a structure lacking hierarchy. (a) Represents the first-level classification with two classes, (b) represents the second-level classification with five classes, and (c) represents the third-level classification with 11 classes.

Figure 5. Proposed hierarchical classification of botnet.

The HCNN model outlined in Figure 5 comprises two components: the backbone, the same as that of the non-hierarchical models, and the hierarchical network. The hierarchical network operates by combining the features acquired at a higher level with those of its ancestor. This combined information is then provided as input to the subsequent layer to acquire features for the descendant nodes. The network comprises 1790 parameters, with the hierarchical network introducing 942 parameters to the backbone. The backbone network produces a root representation $R_0$, which is given by the function in Equation (1) which is the global average pooling of the final convolutional layer of the backbone network, where $X$ is the input features and $\theta_0$ is the parameters of the network.
\[ R_0 = \text{GAP}(F(X, \theta_0)) \] (1)

For the hierarchical network, we first produce an independent representation for the first level, with \( R_1 \), given in Equation (2), where \( W_1 \) is the weight for the first layer.

The representation for the second level, denoted as \( R_2 \), is formulated by performing a matrix multiplication between \( W_2 \) and the concatenation of \( R_0 \) with \( R_1 \), as presented in Equation (3).

\[ R_1 = W_1 \times R_0 = W_1 \times \text{GAP}(F(X, \theta)) = H(X, \theta_1) \] (2)

\[ R_2 = W_2 \times (R_0 \oplus R_1) = G(X, \theta_2) \] (3)

Finally, the representation for the third layer, denoted as \( R_3 \), is expressed in Equation (4), where \( W_3 \) denotes the weight matrix operating on the concatenation of vectors \( R_0 \), \( R_1 \), and \( R_2 \).

\[ R_3 = W_3 \times (R_0 \oplus R_1 \oplus R_2) = T(X, \theta_3) \] (4)

The loss of the network, Equation (5), is calculated by summing the loss of each level, \( \text{loss}_l \), where the losses are calculated using binary cross entropy given in Equation (6) for the first level, and categorical cross entropy as presented in Equation (7) for level 2 and 3.

\[ \text{loss} = \sum_{l=1}^{3} \text{loss}_l \] (5)

where the variable “\( l \)” denotes the hierarchical level within the model, and specifically, there are three hierarchical levels in consideration.

\[ \text{loss}_1 = -\frac{1}{N} \sum_{i=1}^{N} [y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)] \] (6)

where \( N \) is the total number of samples, \( y_i \) is the true label of the \( i \)th sample (either 0 or 1), and \( p_i \) is the predicted probability of the \( i \)th sample belonging to class 1.

\[ \text{loss}_{2/3} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \cdot \log(p_{ij}) \] (7)

where \( N \) is the total number of samples, \( C \) is the number of classes, \( y_{ij} \) is a ground truth label for the \( i \)th sample and belongs to class \( j \), and \( p_{ij} \) is the predicted probability that the \( i \)th sample belongs to class \( j \).

To guarantee the acquisition of accurate category structures, it is essential to introduce a supplementary loss function known as the dependency loss. The dependency loss, \( \text{dloss}_l \), is calculated by Equation (8) as discussed in [24]. This loss function is formulated by assessing whether the model’s predictions conform to the anticipated hierarchical relationships among classes in the classification task. When the predicted classes of consecutive layers deviate from the parent–child relationship within the hierarchy, this loss penalizes the model, intending to promote the acquisition of precise category structures. The definition of the loss function hinges on evaluating whether the model’s predictions align with the expected hierarchical relationships among classes in the classification task.

\[ \text{dloss}_l = -(\text{loss}_{l-1})^{D_{l-1}} \text{loss}_l^{D_l} \] (8)

where \( D_l \) and \( l \) determine if the model output conflicts with the hierarchical structure.
\( D_l = \begin{cases} 1 & \text{if } p_l \neq p_{l-1} \\ 0 & \text{otherwise} \end{cases} \) \hspace{1cm} (9)

\( I_l = \begin{cases} 1 & \text{if } p_l \neq y_l \\ 0 & \text{otherwise} \end{cases} \) \hspace{1cm} (10)

where \( p_l \) and \( y_l \) denote the prediction and actual label at level \( l \), respectively. If the category prediction at level \( l \) aligns with the prediction at level \( l-1 \), \( D \) is set to 0; otherwise, it is set to 1. Similarly, if the prediction at level \( l \) aligns with the actual value at the same level, \( I \) is set to 0; otherwise, it is set to 1.

The final loss Equation (11) is a weighted summation of the loss in \( \text{loss1, loss2, loss3} \) and \( d\text{loss} \), using \( \alpha \) and \( \beta \) as weights of the two losses respectively, where \( 0 \leq \alpha, \beta \leq 1 \).

\[ L(\theta) = \sum_{l=i}^{3} \alpha_i \text{loss}_i + \sum_{l=2}^{3} \beta_i d\text{loss}_i \] \hspace{1cm} (11)

### 3.3. Evaluation Parameters

Precision, recall, F1 score, and accuracy are metrics commonly used to evaluate the performance of classification models [17]. These metrics are calculated based on the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These terms are commonly used in binary classification scenarios, where a model is predicting between two or more classes (positive and negative) [17].

- **Precision**: This is also known as a positive predictive value, which measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives as presented in Equation (12).

\[ P = \frac{TP}{TP + FP} \] \hspace{1cm} (12)

- **Recall**: This is also known as sensitivity or true positive rate, which measures the ability of the model to correctly identify all relevant instances in the dataset. It is calculated as the ratio of true positives to the sum of true positives and false negatives as presented in Equation (13).

\[ R = \frac{TP}{TP + FN} \] \hspace{1cm} (13)

- **F1 score**: This is the harmonic mean of precision and recall, providing a balance between the two metrics as presented in Equation (14). It is particularly useful when there is an uneven class distribution.

\[ \text{F1 Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} \] \hspace{1cm} (14)

- **Accuracy**: This measures the overall correctness of the model by considering both true positives and true negatives. It is calculated as the ratio of correctly predicted instances (TP + TN) to the total number of instances as presented in Equation (15).

\[ \text{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \] \hspace{1cm} (15)

- **Matthews correlation coefficient (MCC)** [26]: This considers both true and false positives and negatives, making it a well-balanced measure suitable for scenarios where class imbalance. Ranging from \(-1\) to \(+1\), the MCC essentially represents a correlation coefficient. A value of \(+1\) indicates a flawless prediction, \(0\) denotes an average random
prediction, and −1 signifies an inverse prediction. The formula to calculate MCC is given in Equation (16).

\[
\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}
\] (16)

- Cohen’s Kappa (k) [27]: This serves as a statistical metric for measuring inter-annotator agreement. This function calculates Cohen’s kappa, a score that quantifies the level of agreement between two annotators in a classification problem. The definition, as presented in Equation (17), involves \( p_o \), the empirical probability of agreement on the label assigned to any sample (the observed agreement ratio), and \( p_e \), the expected agreement when annotators assign labels randomly. Estimating \( p_e \) utilizes a per-annotator empirical prior over the class labels.

\[
k = \frac{p_o - p_e}{1 - p_e}
\] (17)

4. Experimental Results

In this investigation, we conducted two primary experiments: the initial involved training and fine-tuning the hierarchical model, while the second replicated the same procedures for the non-hierarchical model. These experiments aimed to evaluate the performance of the proposed HCNN classification and to assess whether the proposed model outperforms the existing state-of-the-art methods. The optimal hyper-parameters value has been obtained using grid-search, as indicated in Table 2.

<table>
<thead>
<tr>
<th>Hyper-Parameters</th>
<th>Possible Values</th>
<th>Optimal Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation Function</td>
<td>ReLu, Leaky ReLu, tanh</td>
<td>tanh</td>
</tr>
<tr>
<td>Batch size</td>
<td>8, 16, 32, 64, 128</td>
<td>32</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.01, 0.001, 0.0001</td>
<td>0.001</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD, RMSprop, Adam</td>
<td>Adam</td>
</tr>
</tbody>
</table>

Figures 6 and 7 provide a visual representation of the learning curve and confusion matrix associated with the HCNN model. The depicted curve exhibits a notable smoothness, indicating a balanced learning process. Notably, there is an absence of both overfitting and underfitting, crucial aspects in assessing the model’s generalization capabilities. This observation is particularly significant, as it suggests that the HCNN model effectively navigates the trade-off between complexity and simplicity, showcasing its ability to learn intricate patterns within the dataset. An absence of overfitting implies that the model does not excessively tailor itself to the training data, avoiding memorization of specific instances at the expense of broader applicability. On the other hand, the lack of underfitting signifies that the model is not oversimplified to the extent of failing to capture essential patterns within the data. The fact that the HCNN model achieves this equilibrium despite its lightweight design underscores its efficiency and competence in learning and generalizing from the given dataset.
Figure 6. Training curve of HCNN accuracy across different levels: (a) initial level, (b) second level, (c) third level; (d) represents overall training loss.

Figure 7. Confusion matrices for (a) the level with 2 classes, (b) the level with 5 classes, and (c) the level with 11 classes.

The outcome of the proposed approach presented in Figure 8 pertains to the classification performance of the models at the initial hierarchy level, encompassing two categories: “attack” and “normal”. Examining the results outlined in Figure 9, it becomes evident that there is no notable performance improvement at the second level attributable to the hierarchical organization of the model, with both models showcasing comparable scores. Considering these observations, it can be inferred that the hierarchical structure’s impact is not markedly significant at the first level.

Further exploring the hierarchical structure of data during model development offers a primary advantage, particularly in enhancing the performance of the higher levels within the hierarchy. This advantage stems from the fact that the features learned at the initial levels of the hierarchy contribute significantly to distinguishing between classes that may be inherently challenging to separate. This is underscored by the substantial performance improvement observed at the third level hierarchy, as evidenced in Figure 10, where the $F_1$ score for the HCNN stands at 98%, surpassing the non-hierarchical model’s score of 93%. 
The results in Table 3 reveals that the model’s performance remains robust despite the imbalanced dataset, as evidenced by the MCC and k values. Our approach involved utilizing the dataset in its natural form, showcasing the model’s effectiveness under such conditions. Both evaluation metrics consistently yielded scores above 0.99 at each hierarchical level, affirming the proposed model’s robustness and excellent performance, even in the face of an unbalanced dataset. The confusion matrix shown in Figure 7 illustrates the classification performance at each level, revealing minimal misclassifications with the model accurately identifying the majority of instances.
Figure 10. Classification result for the third level in the hierarchy.

Table 3. MCC and k result for each level in the hierarchy.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Level-1 (2-Classes)</th>
<th>Level-2 (5-Classes)</th>
<th>Level-3 (11-Classes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCC</td>
<td>0.992</td>
<td>1.00</td>
<td>0.998</td>
</tr>
<tr>
<td>K</td>
<td>0.992</td>
<td>1.00</td>
<td>0.998</td>
</tr>
</tbody>
</table>

Discussion

The performance of the suggested model closely matches that of the state-of-the-art approach in both two-class and five-class classifications. The classification of a single instance is quick, taking only 9.4 µs, tested on an Apple MacBook Pro M1. Notably, the proposed model stands out for its lightweight design, demanding a mere 6.97 KB in memory. In the case of the third-level 11-class classification within the hierarchy, our model continues to outperform existing works in terms of both performance and computational resource demands.

Comparing our suggested methodology with current state-of-the-art methods allowed for a thorough assessment of its effectiveness, provided in Tables 4–6. In particular, our method was thoroughly tested against prominent techniques [14,17,18]. This thorough comparison study sought to clarify our suggested approach’s unique benefits and functional characteristics with these state-of-the-art works.

Table 4. Comparing the proposed model performance to the existing techniques (2 classes).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree [18]</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Proposed</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5. Comparing the proposed model performance to the existing techniques (5 classes).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree [18]</td>
<td>100</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XGBoost [17]</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Proposed</td>
<td>100</td>
<td>100</td>
<td>99</td>
<td>99</td>
</tr>
</tbody>
</table>
Table 6. Comparing the proposed model performance to the existing techniques (11 classes).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree [18]</td>
<td>100</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>KNN [18]</td>
<td>99.982</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SVM [18]</td>
<td>99.967</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Proposed</td>
<td>100</td>
<td>99</td>
<td>97</td>
<td>98</td>
</tr>
</tbody>
</table>

The CNN architecture proposed in [14] is more intensive compared to the architecture proposed in this work in terms of learnable parameters. Although the performance observed in [18] is encouraging, it is worth noting that the utilization of the SMOTE technique, as highlighted by researchers in [28,29], can potentially generate synthetic examples that are very similar to existing minority class instances, leading to overfitting.

Developing a detection model tailored for BotNet attack detection within the context of SDN-enabled IoT architecture, as depicted in Figure 11, entails consideration of four primary layers: device, network, controller, and application layers [11,30]. At the device layer, sensors and actuators detect and interact with the environment. The network layer includes SDN gateways and routers responsible for data forwarding, controlled by the SDN controller. The controller layer encompasses the SDN controller, which developers program to create IoT services. Typically, the controller layer is utilized to enable high-security protocols.

Implementing lightweight machine learning models for botnet attack detection in IoT devices requires careful selection of efficient algorithms and optimization of their architecture and parameters to minimize resource usage while preserving detection accuracy. As shown in Figure 11 the proposed ML model is deployed on the device layer for analyzing data streams in real-time so that instant decisions are made without relying on centralized processing. The deployment architecture coupled with a very small size of the model, 6.97 KB, leads to faster response time and improved network performance. Alternatively, the model can be deployed on the data plane, encompassing both network and device layers, as discussed in [31–33], allowing for attack detection on devices like gateways and routers.

Figure 11. SDN-enabled IoT architecture and ML model deployment on the device layer.
5. Conclusions

By leveraging the hierarchical arrangement of the data, a model can capitalize on the insights gained at lower levels to better discern intricate patterns or subtle differences at the higher levels. This hierarchical learning process allows the model to develop a better understanding of the data, which in turn improves its ability to differentiate between closely related or inseparable classes. As the model progresses through the hierarchy, it refines its feature representations, thus empowering it to make more informed and accurate predictions, particularly at the levels where class separability is inherently challenging. This holistic approach to hierarchical modeling ultimately contributes to the overall effectiveness and robustness of the machine learning model.

This study introduces HCNN, a hierarchical model utilizing a lightweight CNN, for BotNet attack detection. With a small number of parameters, this model boasts efficient memory usage and requires minimal running time, rendering it well-suited for IoT devices. It achieved commendable results, surpassing 0.99 for both MCC and k values at each level. This indicates the robustness of the model despite the presence of an unbalanced dataset. Notably, the $F_1$ scores exhibited remarkable performance across all levels, with a significant improvement observed at the third level featuring 11 classes, attaining an $F_1$ score of 0.98. This underscores the hierarchical model’s efficacy in enhancing classification, especially at deeper levels, by leveraging features from the more distinguishable lower levels compared to the higher ones.

The effectiveness of the proposed model has only been evaluated using a single dataset. To ensure its performance in diverse scenarios, it is crucial to train and test the model with a variety of datasets. Our forthcoming plans include assessing the model’s robustness by exposing it to different datasets collected under various circumstances. The outcomes presented in our paper reflect the results of experiments conducted with our classification model using this particular dataset. Furthermore, our future efforts involve testing the model under real-world conditions, especially within an SDN-orchestrated IoT environment.

Author Contributions: Conceptualization, W.G.N.; methodology, W.G.N.; validation, F.S., T.G.D., D.W.F. and H.M.M.; writing—original, W.G.N.; writing—review and editing, T.G.D., F.S., D.W.F. and H.M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [https://research.unsw.edu.au/projects/bot-iot-dataset].

Conflicts of Interest: The author declares no conflicts of interest.

References


---

**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.