The Lightweight Botnet Detection Model Based on the Improved UNet

Chengjie Li[1,2], Yunchun Zhang[1,2], Zixuan Li[1,2], Fan Feng[1,2], Zikun Liao[1,2], and Xiaohui Cui[2,3(✉)]

[1] School of Software, Yunnan University, Kunming, China
yczhang@ynu.edu.cn, zixuanli@mail.ynu.edu.cn
[2] Research Center of Cyberspace, Yunnan University, Kunming, China
[3] School of Cyber Science and Engineering, Wuhan University, Wuhan, China
xcui@whu.edu.cn

Abstract. Botnet detection tasks in many network devices require deployment of a large number of detection models. Deep learning-based Botnet detection models are big and resource-intensive. Besides, the UNet is primarily used for two-dimensional inputs but with higher complexity. This paper presents a One-Dimensional UNet (1D-UNet) based on one-dimensional feature vectors generated to design a lightweight detection engine. Second, we propose a One-Dimensional Lightweight UNet (1DL-UNet) by combining the 1D-UNet with depthwise separable convolution to reduce the model's complexity. Finally, we reduce the number of packets for Botnet detection based on our observation that the first packet with an effective payload in a network session plays the most important role in detection. The experiments show that the 1DL-UNet outperforms other models with 99.66% accuracy and is 12 times smaller than one-dimensional MobileNet. Meanwhile, the designed 1DL-UNet is 4 times smaller than the 1D-UNet. Furthermore, it is observed that 4 packets are enough to achieve satisfactory Botnet detection while only 1 packet with effective payload is possible with 99.26% accuracy in the 1DL-UNet.

Keywords: Botnet detection · Convolutional neural networks · One-dimensional convolution · UNet

1 Introduction

The deployment of network devices is remarkably increased in the areas of industry, health care, smart home, and smart city. The number of devices integrated into the Internet has sharply increased and it is estimated that there will be 75.44 billion of IoT devices by 2025 [1]. IoT device is mainly composed of low-cost devices with limited resources by combining physical devices with digital intelligent networks. The whole network is highly heterogeneous and dynamic. The nature of limited resource on a end device and the large volume of network

S. Goel et al. (Eds.): ICDF2C 2022, LNICST 508, pp. 241–255, 2023.
https://doi.org/10.1007/978-3-031-36574-4_14

traffic among inter-connected devices on the Internet makes those devices fail to implement highly complex security tasks. Meanwhile, many vulnerabilities are born in nature within a network and are used for launching illegal activities to control, subvert and destroy network devices. A typical attack scenario is Botnet where attackers can manipulate some devices to launch DDoS (Distributed Denial-of-Service) attack. In a typical Botnet attack, once a end device is controlled by an attacker, the attacker will then enlarge its attacking speed and target more devices by using Command-and-Control ($C\&C$) communications. The attacking effect can be greatly enhanced then. Meanwhile, many Botnets are using BaaS (Botnet as a Service) to provide network services. This greatly helps the botmaster to control and manipulate attacks while hiding from being detected. Therefore, Botnets are flourishing on the Internet and the scale of attacks is increasingly large, such as that in Mirai [2]. In Mirai, a large number of network devices are manipulated to launch DDoS against the provider's DNS (Domain Name Server).

Deep neural networks are trained and deployed for Botnet detection recently [3]. But the trained neural networks are usually computing-intensive and resource-intensive. To solve the Botnet detection problem on the Internet with the aim of both improving the detection accuracy and simplifying the data processing procedure, this paper makes the following contributions.

(1) Based on the combination of the network packet transmission state and transmission contents, we improved the UNet [4] for Botnet detection based on network traffic. By converting the network traffic into bytes streams and used as inputs, the One-Dimensional UNet (1D-UNet) convolutional neural network is proposed. This 1D-UNet achieves 0.42% improvement on accuracy than the best model in [5].
(2) As network devices are usually power limited, we improve 1D-UNet and propose the One-dimensional Lightweight UNet (1DL-UNet) based on the design methodology and mechanism in MobileNet [6]. The experimental results show that the 1DL-UNet achieves comparable detection accuracy while reducing its parameter volumes to 8% of the 1D-MobileNet and reducing the computational cost to 22% of the 1D-MobileNet. The 1DL-UNet is also applicable for Botnet detection on end devices on the Internet.
(3) We propose a method to reduce the number of network packets required to detect the Botnet with comparable performance. The proposed method relies on the extraction of the first network packet with effective payloads. The experiments show that only one network packet within a session is enough for effective Botnet detection. Two critical factors that have a direct impact on the detection performance are also analyzed, including the number of packets selected and the length of packets extracted (in bytes).

2 Related Works

Botnets are quickly evolved with more powerful destruction against both the Internet and IoT. In recent years, deep learning-based detection models are

popular for Botnet detection by modeling it as an anomaly detection problem. The anomaly detection-based methods are primarily based on the observations of communication patterns from the network traffic [7]. BotHunter [8] achieves Botnet detection based on the sequential communication traffic in the periods of bot broadcasting and infection. BotMiner [9] and BotSniffer [10] are mainly based on measuring the similarity of their behavior. Other features extracted based on the TCP connection and network traffic logs are also applicable for anomaly detection [11,12].

The deep learning-based Botnet detection methods are more effective than anomaly detection-based models. Many deep neural networks are trained based on network traffic data, such as RNN (Recurrent Neural Network) [13], Bi-LSTM (Bi-directional Long Short-Term Memory) [14], deep auto-encoder [15], etc. Some benchmark datasets for Botnet detection are proposed and ISCX Botnet [16] is the most popular one. To enhance the detection performance, Hosseini et al. [17] addressed the problem of weak correlation of statistical features of network traffic by using a two-stage feature fusion method. The proposed solution uses CNN for spatial logical correlation of statistical features in the first stage and LSTM for temporal logical correlation of the proposed features in the second stage to ensure a high detection accuracy even when using statistical features with weak correlation. The BotCathcer [18] proposed a detection method of byte sequences and statistical features as combined features. Both imaged byte sequences and statistical features are input into CNN and LSTM respectively. Then, the output nodes of the two networks combine the features for subsequent feature input. As some botnet samples are small, Zou et al. [5] investigated the application of GAN (Generative Adversarial Network) in image recognition and proposed a DCGAN-based botnet detection model. The proposed method can effectively expand the number of samples and improve the detection capability of the model while achieving the highest detection accuracy of 99.23%.

Although deep learning models are featured by their efficient automatic feature extraction and satisfactory performance for Botnet detection, most of them are computation-intensive and unsuitable for end devices in IoT especially for micro devices based on Micro-Controller Units (MCU) [19]. For deployment into low-energy environments, the authors in [20] combine the BNN (Binarized Neural Network) with federated learning for Botnet detection, but the accuracy is only 94.5%. Thus, designing a lightweight neural network for Botnet detection is challenging. This paper aims to design a lightweight model that is not only energy-saving but also comparable to the previous methods on all performance metrics.

3 Methodology

Many state-of-the-art Botnet detection models, especially deep neural network-based, are applicable on the Internet. But a typical deep neural network is too big to apply on devices with limited resources. A successful Botnet detection model for IoT devices should be small in size. To design a lightweight detection

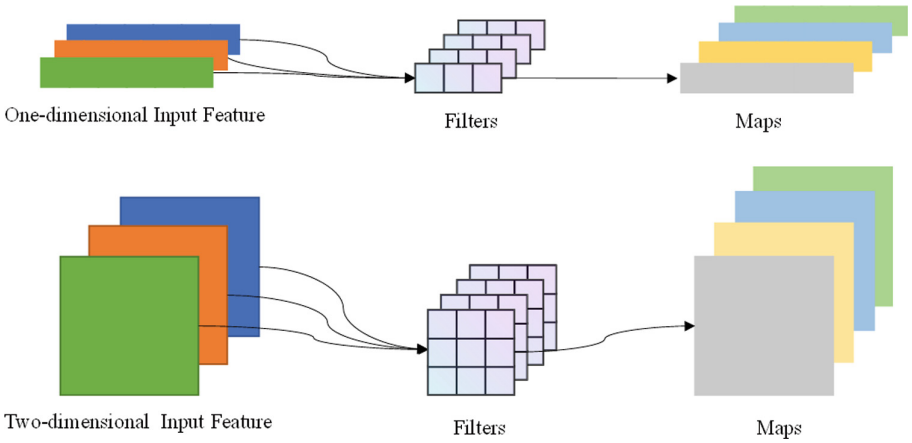

Fig. 1. The differences between the one-dimensional and two-dimensional Convolutional Neural Networks.

engine in Internet, this paper optimizes the existing method from two sides: model optimization and inputs reduction.

3.1 1D-UNet for Botnet Detection

Deep learning models are widely deployed for botnet detection based on traffic gray-scale images. Gray-scale images generated from either traffic flow byte sequence or features extracted from the network traffic flows are two-dimensional. Thus, Two-Dimensional Convolutional Neural Networks (2D-CNN) are introduced for Botnet detection because those models are commonly used for image classification and segmentation. The 2D-CNN is good at extracting spatial structure patterns and analyzing the space logic within a two-dimensional space. On the contrary, One-Dimensional CNN (1D-CNN) is good at extracting useful features from a short or fixed-length segment. The filter in a 1D-CNN usually slides in a unidirectional way and the convolutional operation is considered as a feature distillation process without any feature ordering or feature selection. All in all, the differences between 1D-CNN and 2D-CNN are shown in Fig. 1.

Based on the basic configurations of network and the requirement of training a lightweight Botnet detection model on end devices, this paper designed a small convolutional neural network, called One-Dimensional UNet (1D-UNet). The designed model is based on the improvements to UNet [4] that featured with its high performance on two-dimensional inputs and is commonly used for image classification and segmentation. While Botnet attacks are usually with a high volume of network packets, the designed 1D-UNet uses network packets, including packet header and packet payload, as inputs.

We define a series of network packets as a **Network flow**, abbreviated as *NetFlow*. A *NetFlow* is defined as a logical unit of packets denoted by a five

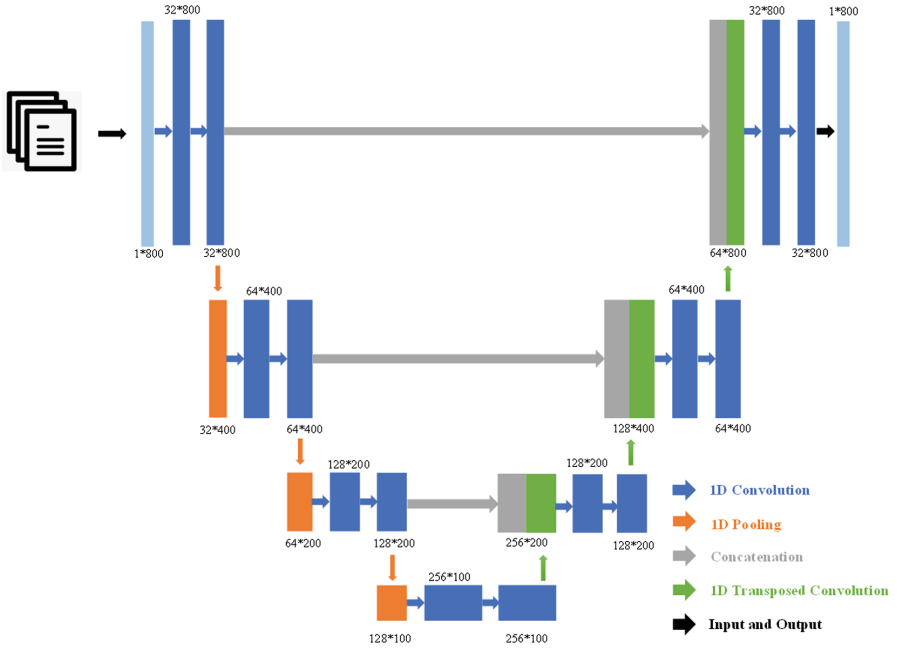

Fig. 2. The system architecture of the 1D-UNet for Botnet detection.

elements tuple as $<SrcIP, SrcPort, DstIP, DstPort, Protocol>$, where Src and Dst means the source node and the destination node, respectively. The *NetFlow*, as a uni-directional processing method of packets, is highly effective for network anomaly detection. However, Botnet is highly characterized by its bi-directional communications. We combine two *NetFlows* with the same *IP* and *Port* into a single *session*. By extracting a segment of the *session* as input x, the designed UNet is defined as $y_i = F_\theta(x_i)$, where $y_i \in \{0, 1\}$ which means the given input is classified as Botnet when $y_i = 1$. Each input x_i is extracted from X^d where d means the length of the extracted segment in bytes. Once trained and optimized, the 1D-UNet is with the best parameter value θ. The whole process from input x into the 1D-UNet to output $\hat{y} \in \{0, 1\}$ is denoted as a mapping by $f : x \to \hat{y}$. While inputs are bytes extracted from the *NetFlow*, the whole mapping process aims at minimizing the loss function $\mathcal{L}(\theta, x, y)$ where we choose binary cross-entropy and define the objective function as:

$$\mathcal{L}(\Theta, x, y) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))) \tag{1}$$

where, N means the total numbers of predicted classes and $N = 2$ in this paper as a typical binary classification problem.

The designed 1D-UNet is mainly composed of a U-type structure with encoding and decoding. The encoding focuses on feature extraction from input byte sequences based on convolutional operations and down-sampling. By applying up-sampling to the inputs from the encoding component, the decoding restores a feature to its original size. Then, the last fully connected layer output the predicted labels for Botnet detection. As the byte sequences used in this paper are discrete and differ from two-dimensional images, the designed 1D-UNet is configured with less convolutional layers for efficient processing of byte sequences. The network architecture of the 1D-UNet is shown in Fig. 2.

3.2 1DL-UNet for Botnet Detection

The 1D-UNet uses multiple convolutional blocks with twice one-dimensional convolutional operations. The classification accuracy is enhanced but the number of parameters and the computation cost increased simultaneously. To reduce the number of parameters to a reasonable level, the Depthwise Separable Convolution (DSC) in MobileNet [6] is introduced. We then propose a One-Dimensional Lightweight UNet (1DL-UNet) for Botnet detection in Internet.

The 1DL-UNet optimizes the original 1D-Unet from two sides. First, the one-dimensional convolutional operations in the original convolution block are substituted for one-dimensional depthwise separable convolution (1D-DSC). Second, the stride of the convolutional operations in the process of the second depthwise separable convolution block is enlarged from 1 to 2, replacing the original max pooling by reducing the dimension during feature learning. Based on the above improvements, the architecture of the 1DL-UNet is shown in Fig. 3.

The 1D-DSC, as a decomposed form of the standard one-dimensional convolutional operation, is composed of depthwise convolution and pointwise convolution as shown in Fig. 4. The standard one-dimensional convolution takes $D_{in} \times M$ as the input feature mapping F and output $D_{out} \times N$ as the output feature mapping G, where D_{in} represents the size of the input feature vector. M means the number of channels of the input feature vector. D_{out} represents the width and height of the output features. N means the number of channels of the output feature vector. Then, the one-dimensional convolutional kernel K is defined as:

$$K = D_k \times M \times N \tag{2}$$

where, D_k means the dimension on features by convolutional kernel. If we set both the stride and padding as 1, the output feature mapping is computed by the standard convolution as:

$$G_{k,n} = \sum\nolimits_{i,m} K_{i,m,n} \cdot F_{k+i-1,m}. \tag{3}$$

where, m and n represent the input and output channel, k represents the size of the convolution kernel, and i represents the beginning position of the convolution operation on the input vector.

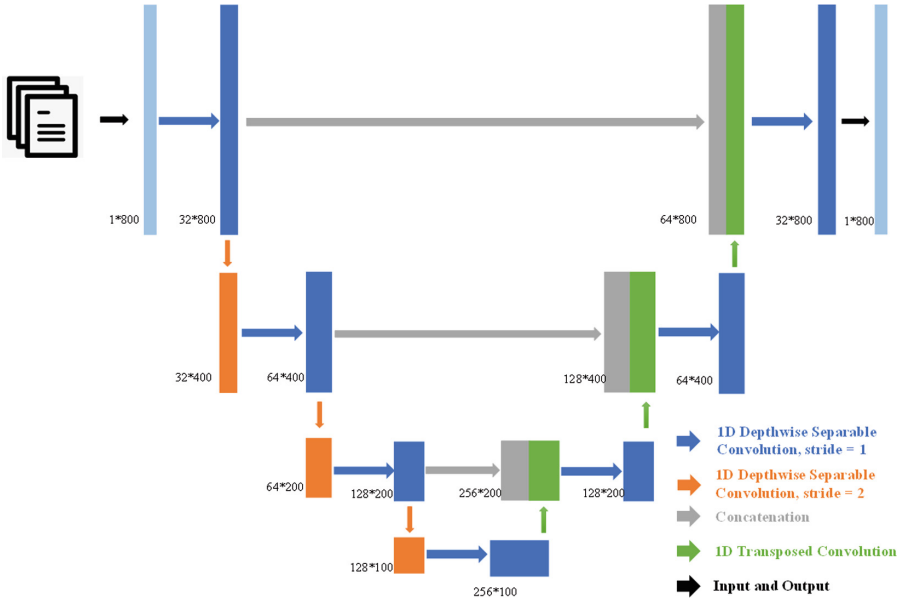

Fig. 3. The architecture of the 1DL-UNet for Botnet detection.

Then, the number of parameters $Params_{Conv}$ and the computational cost $FLOPs_{Conv}$ in floating point operations can be computed as:

$$Params_{Conv} = D_k \cdot M \cdot N \tag{4}$$

$$FLOPs_{Conv} = D_k \cdot M \cdot D_{out} \cdot N \tag{5}$$

As shown in Eq. (4) and (5), the computation cost is determined by the number of input and output channels, kernel size, and feature mappings. To reduce the cost to a reasonable level, the designed 1DL-UNet decomposes the standard convolutional operations based on diminishing the dependence between the output channels and the size of the convolutional kernel. We substitute the filter and combination operations in the standard convolution process with depthwise convolution and pointwise combination in 1DL-UNet, respectively. The one-dimensional depthwise convolution applies a filter on each input channel as:

$$G'_{k,m} = \sum_i K'_{k,m} \cdot F_{k+i-1,m} \tag{6}$$

where, K' is a depthwise convolutional kernel with $D_K \times D_K \times M$ in size. The m-th filter in K' is applied on the m-th channel in F to generate the m-th channel on the output feature mapping in G'. The depthwise convolution is designed as shown in Fig. 5.

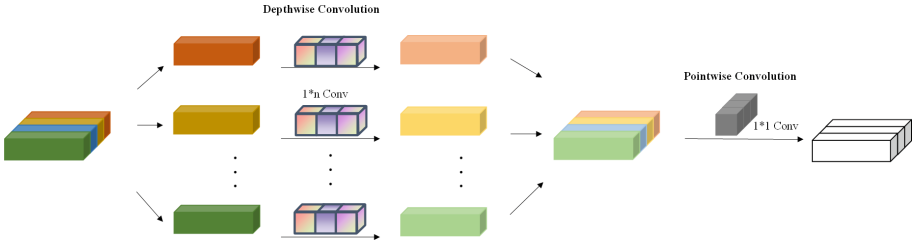

Fig. 4. The one-dimensional (1D) depthwise separable convolution.

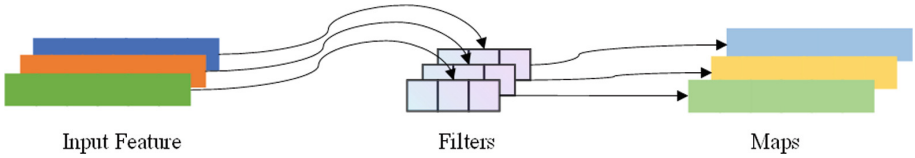

Fig. 5. The detailed implementation of the depthwise convolution.

Based on the above analysis, the number of parameters $Params_{DC}$ and the computation cost $FLOPs_{DC}$ in the depthwise convolution are computed as:

$$Params_{DC} = D_K \cdot M \tag{7}$$

$$FLOPs_{DC} = D_K \cdot M \cdot D_{out} \tag{8}$$

As the depthwise convolution only achieves filtering operations to the input feature mappings, it fails to combine all filtered feature maps into an integrated output. To solve this issue, we introduce an additional linear combination with 1×1 convolutional operation in a pointwise convolutional layer, as shown in Fig. 6.

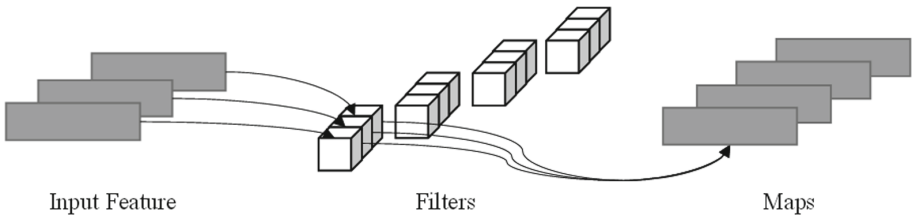

Fig. 6. The detailed implementation of the pointwise convolution.

Based on the above design, the number of parameters $Params_{PC}$ and the computation cost $FLOPs_{PC}$ in the pointwise convolution are computed as:

$$Params_{PC} = M \cdot N \tag{9}$$

$$FLOPs_{PC} = M \cdot D_{out} \cdot N \tag{10}$$

By combining the depthwise convolution with pointwise convolution, the overall parameter volume and computation cost in the 1D-DSC are computed as:

$$Params_{DSC} = D_k \cdot M + M \cdot N \tag{11}$$

$$FLOPs_{DSC} = D_K \cdot M \cdot D_{out} + M \cdot D_{out} \cdot N \tag{12}$$

Finally, the ratio of both the number of parameters $Params_{ratio}$ and the cost computation $FLOPs_{ratio}$ between the depthwise separable convolution and the standard convolution are computed as:

$$Params_{ratio} = \frac{D_K \cdot M + M \cdot N}{D_K \cdot M \cdot N} = \frac{1}{N} + \frac{1}{D_K} \tag{13}$$

$$FLOPs_{ratio} = \frac{D_K \cdot M \cdot D_{out} + M \cdot D_{out} \cdot N}{D_K \cdot M \cdot D_{out} \cdot N} = \frac{1}{N} + \frac{1}{D_K} \tag{14}$$

The degree of reduction in the number of parameters and computations contained in depthwise separable convolution and the number of output channels and convolution during convolution operations are as shown in Eqs. (13) and (14). The kernel size is related. When the output channel and the convolution kernel are large, the content of parameters and computation in the depthwise separable convolution is more different from the original standard convolution. The deep neural network model obtained by introducing the depthwise separable convolution meets the goal of botnet detection on the Internet based on network traffic data.

4 Experiments and Result Analysis

4.1 Configurations, Data Processing and Evaluation Metrics

All models trained in this paper are running on Ubuntu with 32GB memory, Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40 GHz, and TITAN XP. All models are written by Pytorch and accelerated with Adam. We set $batch\_size = 64$, $epoch = 40$ and $learning\_rate = 0.001$.

The benchmark ISCX Botnet dataset [16] is chosen. This dataset contains 16 types of botnet attacks and benign traffic. The original data are stored in *.pcap* files. We organized the network traffic into *NetFlow* with $P = p_1, p_2, ..., p_n$ while $p_i = (f_i, s_i, t_i)$, where f_i represents the *NetFlow* information extracted, s_i means the size of this network flow and t_i means the beginning time when communication happens. When *NetFlow* differentiate the communications in directions, we combine them into a network session defined as $Session_j = p_1, p_2, ..., p_k$ and all flows are organized sequentially according to their happening times where $t_1 < t_2 < \cdots < t_k$.

To reduce the number of packets for Botnet detection with both 1D-UNet and 1DL-UNet, two strategies denoted as $Exp1$ and $Exp2$ are applied. We extract

the first packet in a session for analysis in $Exp1$. In $Exp2$, if a session contains effective payloads, we choose the first packet with an effective payload as the starting point to extract a given length of bytes for later analysis. Otherwise, starting with the first packet of the session. When the number or length of the data packets is insufficient, we use $0x00$ for padding. Finally, all samples extracted are randomly partitioned into the training set, validation set, and testing set with $3:1:1$ in ratio.

All models are compared under the same performance metrics, including accuracy, precision, recall and F1-score, as shown in Eqs. (15)–(18).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{15}$$

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{18}$$

where $TP(TruePositive)$ represents the number of Botnet traffic that can be correctly detected, $TN(TrueNegative)$ represents the number of benign traffic that can be correctly detected, $FP(FalsePositive)$ represents the number of benign traffic that mistakenly classified as Botnet, and $FN(FalseNegative)$ represents the number of Botnet traffic that mistakenly classified as benign.

4.2 Results and Analysis

4.2.1 Number of Packets and Performance

The number of packets and the size of each packet varied remarkably among different network communication sessions. The statistical features of packets in the ISCX dataset are summarized in Table 1. As shown in Table 1, the number of packets in a session varies greatly. In our experiments, we set the size of the extracted data as 100. Considering the median value for the number of packets is 6, we tested whether the number of packets will seriously affect the final performance and set different values for this parameter. The results under both the 1D-UNet and the 1DL-UNet are shown in Table 2 and 3.

Based on the performance results under both the 1D-UNet and 1DL-UNet as shown in Table 2 and 3, we made the following observations.

(1) Under the $Exp1$ where we extract packets from the beginning of a network session for analysis, both 1D-UNet and 1DL-UNet achieve the best detection accuracy with 99.65% and 99.66% when 8 packets are extracted for analysis. When more than 4 packets are used, the detection accuracy keeps stable under both 1D-UNet and 1DL-UNet with minor fluctuations as 0.08% and 0.10%.

Table 1. Basic statistical patterns in the ISCX Botnet dataset

Statistical Indicators	Quantity
Total number of sessions	373,107
Average number of packets in a session	36
Maximum number of packets in a session	944,228
Minimum number of packets in a session	1
Mode of packets for sessions	6
Median of packets for sessions	6

Table 2. The performance of two models under different settings with the number of packets extracted for learning on *Exp*1.

No. of packets	1D-UNet				1DL-UNet			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
1	96.81	95.64	96.89	96.26	96.75	96.65	95.82	96.23
2	99.28	99.11	99.22	99.16	99.26	99.30	98.98	99.14
4	99.58	99.54	99.49	99.51	99.63	99.61	99.52	99.56
6	99.59	**99.59**	99.47	99.53	99.61	99.52	99.58	99.55
8	**99.65**	99.50	**99.67**	**99.59**	**99.66**	99.54	**99.68**	99.61
10	99.61	99.49	99.59	99.54	99.62	99.54	99.58	99.56
12	99.63	99.49	99.65	99.57	99.57	99.59	99.40	99.50
14	99.59	99.47	99.59	99.53	99.56	99.49	99.49	99.49
16	99.57	99.51	99.49	99.50	99.61	**99.62**	99.48	**99.69**

(2) Under the *Exp*2 where we extract packets from the first packets with effective payload in a network session for analysis, both 1D-UNet and 1DL-UNet achieve the best detection accuracy with 99.62% and 99.63% when 8 packets are extract for analysis. There is minor performance degradation with 0.03% on average when compared with that under the *Exp*1. The main reason is that some Botnet attacks are mainly DDoS attacks without sending any effective payload.

(3) When both models are tested under two data extraction methods, including *Exp*1 and *Exp*2, it is shown that the detection accuracy under *Exp*2 achieves the best performance than *Exp*1 with 2.69% improvements when using 1 packet for analysis. When more than 2 packets are used, the detection performance quickly converges and remains stable.

(4) When applied on end devices with limited resources and computing power, both 1D-UNet and 1DL-UNet are applicable for Botnet detection with only 4 packets for analysis. Under some critical situations, a satisfactory detection accuracy with only 1 or 2 packets for analysis is possible. Therefore, the designed 1D-UNet and 1DL-UNet are not only lightweight but also effective for Botnet detection on the Internet and IoT.

Table 3. The performance of two models under different settings with the number of packets extracted for learning on *Exp*2.

No. of packets	1D-UNet				1DL-UNet			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
1	99.19	98.92	99.18	99.05	99.26	98.92	99.35	99.14
2	99.34	99.14	99.32	99.23	99.38	99.31	99.24	99.27
4	99.54	99.49	99.44	99.47	99.58	99.46	99.55	99.50
6	99.61	**99.59**	99.49	99.54	99.57	99.54	99.46	99.50
8	**99.62**	99.51	**99.59**	**99.55**	**99.63**	99.51	**99.63**	**99.57**
10	99.47	99.41	99.36	99.38	99.56	99.47	99.50	99.48
12	99.58	99.55	99.46	99.51	99.63	**99.56**	99.58	99.57
14	99.49	99.33	99.49	99.41	99.62	99.56	99.56	99.56
16	99.56	99.53	99.46	99.49	99.61	99.50	99.59	99.55

4.2.2 Number of Parameters and Computation Cost

The top goal of our design is a compressed neural network with satisfactory performance on Botnet detection on the Internet. When configured with the best parameter as shown in the former section, we compared our models with the One-Dimensional Mobile (1D-MobileNet) based on the MobileNet [6]. All three models are compared under two different settings, including *Exp*1 with 8 packets and *Exp*2 with 1 packet, as shown in Table 4 and 5, respectively. Based on the results in Table 4 and 5, we made the following observations.

Table 4. Performance comparison among three models under *Exp*1 with 8 packets for analysis.

Method	Accuracy	Precision	Recall	F1-score	Params(M)	FLOPs(G)
1D-UNet	99.65	99.50	99.67	99.59	0.716	0.141
1DL-UNet	99.66	99.54	**99.68**	99.61	**0.245**	**0.056**
1D-MobileNet	**99.68**	**99.59**	99.65	**99.62**	3.178	0.258

Table 5. Performance comparison among three models under *Exp*2 with 1 packet for analysis.

Method	Accuracy	Precision	Recall	F1-score	Params(M)	FLOPs(G)
1D-UNet	99.22	**98.94**	99.23	99.08	0.715	0.025
1DL-UNet	99.26	98.92	99.35	99.14	**0.243**	**0.007**
1D-MobileNet	**99.28**	98.94	**99.38**	**99.16**	2.434	0.046

(1) All models achieve satisfactory performance on accuracy as higher as 99.6%. When using only 1 packet for analysis, the detection accuracy shows 0.4% degradation when compared with the setting under *Exp*1 with 8 packets. However, the number of packets extracted is greatly reduced and thus more

suitable for end devices in Internet. The computation cost under the 1DL-UNet with $Exp2$ is only 12.5% of that under the 1D-UNet with $Exp1$ on 8 packets.

(2) The number of parameters can be greatly compressed. As a lightweight model, the 1D-MobileNet is smaller than MobileNet [6]. However, the 1D-MobileNet is 4 times bigger than 1D-UNet and 12 times bigger than 1DL-UNet in the number of parameters. Meanwhile, the computation cost of the 1D-MobileNet is 1.5 and 4 times higher than that of the 1D-UNet and 1DL-UNet, respectively.

4.2.3 Performance for Botnet Detection

The designed 1DL-UNet is then compared with other models for Botnet detection on the Internet, including ResNet-DCGAN [5], BiLSTM-GAN [5], Rule Induction [12], BNN [20] and LSTM-CNN [17]. The results are shown in Table 6. Based on the results, we made the following observations.

Table 6. Performance comparison with similar models for Botnet detection.

Method	Accuracy	Precision	Recall	F1-score
ResNet+DCGAN [5]	99.23	99.46	99.29	99.37
BiLSTM+GAN [5]	85.51	91.55	82.38	86.72
Rule Induction [12]	98.8	98.7	98.8	99.4
BNN [20]	94.5	94.5	76.6	84
LSTM-CNN [17]	99	98	**100**	–
1DL-UNet+Exp1(8 packets)	**99.66**	**99.54**	99.68	**99.61**
1DL-UNet+Exp2(1 packet)	99.26	98.92	99.35	99.14

(1) The 1DL-UNet achieves the best performance on all metrics when compared with other models. Even when tested under $Exp2$ with one packet for analysis, the 1DL-UNet is 0.03% better than ResNet-DCGAN [5] on accuracy. The ResNet-DCGAN takes the beginning 1,024 bytes from the network flows for analysis by converting them into images. The generated images are usually failed to preserve the typical communication logic relations. When compared with the ResNet-DCGAN, the 1DL-UNet is not only smaller but also good at extracting network communication logic based on the combination of both network communication states and contents included.

(2) Four models, including BiLSTM-GAN [5], Rule Induction [12], BNN [20] and LSTM-CNN [17], are mainly based on a separated statistical feature extraction process on network packets. Therefore, the detection performance is highly affected by the quality of the feature processing method. As the majority of the feature extraction works are done manually, the performance of the trained model is thus seriously damaged. Thus, the 1DL-UNet outperforms others on the whole. More importantly, the byte sequences extracted

from the raw network packet are effective for Botnet detection instead of using discrete statistical features.

(3) When compared with BNN [20] that mainly deployed on the end device in Internet, the 1DL-UNet achieves 4.76% higher detection accuracy while introducing lower parameters and computation cost. Above all, the 1DL-UNet is applicable on either Internet servers or end devices with limited resources.

5 Conclusions

Designing a high-performance intrusion detection system is challenging because of the tedious configurations and heterogeneous devices on the Internet. Some state-of-the-art solutions are mainly server-oriented and expose the end device to attackers. The Botnet, as a popular attack dominated on the Internet and the Internet-of-Things, is hard to defend without all devices' cooperation in the whole network. Therefore, this paper presents a deep learning-powered Botnet detection model applicable for both servers and end devices. Two models, including 1D-UNet and 1DL-UNet, are demonstrated to be effective on Botnet detection with limited resources.

Based on our work, future research works are possible in the following directions. First, it is possible to distribute a compressed model from the server to the end device instead of training a lightweight model on the end device. But both the distribution process and the compressed model are vulnerable against adversarial attackers. Second, most of the existing solutions are demonstrated to be effective in detecting Botnet except the source tracing especially when DDoS involves. Third, both lightweight models and compressed models are vulnerable to adversarial attacks. Thus, all Botnet detection models should be secured and enhanced.

References

1. Nobakht, M., Sivaraman, V., Boreli, R.: A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. In: 11th International Conference on Availability. Reliability and Security (ARES), pp. 147–156. IEEE, Salzburg (2016)
2. Antonakakis, M., et al.: Understanding the mirai botnet. In: 26th USENIX Security Symposium, pp. 1093–1110. USENIX Association, Vancouver (2017)
3. Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q.V., Padannayil, S.K., Simran, K.: A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities. IEEE Trans. Ind. Appl. **56**(4), 4436–4456 (2020)
4. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
5. Zou, F., Tan, Y., Wang, L., Jiang, Y.: Botnet detection based on generative adversarial network. J. Commun. **42**(7), 95–106 (2021)

6. Howard, A.G., Zhu, M., Chen, B.: MobilenEts: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
7. Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S.: Botnet detection based on traffic behavior analysis and flow intervals. Comput. Secur. **39**, 2–16 (2013)
8. Gu, G., Porras, P.A., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: detecting malware infection through IDS-driven dialog correlation. In: 16th USENIX Security Symposium, vol. 7, pp. 167–182. USENIX Association, Boston (2007)
9. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: 17th USENIX Security Symposium, pp. 139–154. USENIX Association, San Jose (2008)
10. Gu, G., Zhang, J., Lee, W.: BotSniffer: detecting botnet command and control channels in network traffic. In: Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, pp. 1–19 (2008)
11. L. Bernaille, R. Teixeira, K. Salamatian: early application identification. In: ACM CoNEXT Conference, Lisbon, pp. 1–12 (2006)
12. Mahardhika, Y.M., Amang, S., Barakbah, A.R.: An implementation of Botnet dataset to predict accuracy based on network flow model. In: 6th International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), pp. 33–39. IEEE, Surabaya (2017)
13. Torres, P., Catania, C., Garcia, S., Garino, C.G.: an analysis of recurrent neural networks for botnet detection behavior. In: 2016 IEEE Biennial Congress of Argentina (ARGENCON), pp. 1–6. IEEE, Buenos Aires (2016)
14. McDermott, C.D., Majdani, F., Petrovski, A.V.: Botnet detection in the internet of things using deep learning approaches. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE, Rio de Janeiro (2018)
15. Meidan, Y., Bohadana, M., Mathov, Y., Shabtai, A., Breitenbacher, D., Elovici, Y.: N-BaIoT: network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput. **17**(3), 12–22 (2018)
16. Beigi, E.B., Jazi, H.H., Stakhanova, N., Ghorbani, A.A.: Towards effective feature selection in machine learning-based botnet detection approaches. In: 2014 IEEE Conference on Communications and Network Security (CNS), pp. 247–255. IEEE, San Francisco (2014)
17. Hosseini, S., Nezhad, A.E., Seilani, H.: Botnet detection using negative selection algorithm, convolution neural network and classification methods. Springer Sci. Bus. Media Deutschland GmbH **13**(1), 101–115 (2018)
18. Wu, D., Fang, B., Cui, X., Liu, X.: BotCatcher: botnet detection system based on deep learning. J. Commun. **39**(8), 18–28 (2018)
19. Lin, J., Chen, W.M., Lin, Y., Cohn, J., Gan, C., Han, S.: MCUNet: tiny deep learning on IoT devices. In: 34th Conference on Neural Information Processing Systems (NeurIPS), vol. 33, pp. 11711–11722. Curran Associates Inc., Vancouver (2020)
20. Qin, Q., Poularakis, K., Leung, K.K., Tassiulas, L.: Line-speed and scalable intrusion detection at the network edge via federated learning. In: 2020 IFIP Networking Conference and Workshops, pp. 352–360. IEEE, Paris (2017)