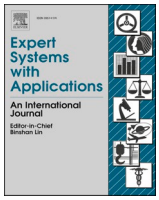
Expert Systems With Applications 238 (2024) 122045

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



A lightweight IoT intrusion detection model based on improved BERT-of-Theseus

Zhendong Wang [a], Jingfei Li [a,*,1], Shuxin Yang [a], Xiao Luo [b], Dahai Li [a], Soroosh Mahmoodi [c]

[a] School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, Jiangxi, China
[b] School of Electrical Engineering and Automation, Jiangxi University of Science and Technology, Ganzhou 341000, Jiangxi, China
[c] Yancheng Teachers University, Yancheng 224000, Jiangsu, China

ARTICLE INFO

Keywords:
Intrusion detection
Internet of Things system
Knowledge distillation
Transformer

ABSTRACT

The proliferation of Internet of Things (IoT) technology has resulted in an increase in security vulnerabilities associated with the interconnectivity of IoT devices. As a result, there is a need for intrusion detection mechanisms that can effectively detect attacks on IoT security vulnerabilities. However, due to the resource constraints of IoT deployment devices, intrusion detection schemes must be customized to meet the specific demands of the IoT environment. In this study, we propose a knowledge-distillation-based IoT intrusion detection model named BT-TPF, which is capable of detecting network attacks encountered by IoT devices in an IoT environment with limited computing resources. The proposed BT-TPF model leverages a Siamese network for feature dimensionality reduction of complex high-dimensional network traffic data. Additionally, it employs a large-scale Vision Transformer as a teacher model to guide a small-scale Poolformer model during training, before deploying the trained Poolformer model as a classifier to detect network intrusion traffic. Through knowledge distillation, the final small model obtained in this paper only requires a minimum of 788 parameters, reducing the number of parameters by approximately 90% compared to the large model before knowledge distillation, while maintaining high detection accuracy. Experimental results show that the BT-TPF model achieves over 99% accuracy on both the CIC-IDS2017 and TON_IoT datasets. Furthermore, it exhibits significant advantages compared to traditional Deep Learning methods and recent state-of-the-art models, as evidenced by various evaluation metrics.

1. Introduction

With the rapid proliferation of Internet of Things (IoT) technology, its applications in various social services, such as smart homes, healthcare, education, and transportation, have expanded significantly (Al-Garadi et al., 2020). IoT devices are now engaging in open and massive communication with millions of other devices across the world, which has rendered them increasingly attractive to malicious actors (Stoyanova et al., 2020). In recent years, there has been a significant surge in attacks targeting IoT devices. These include attacks on wireless webcams, enabling hackers to access surveillance cameras and breach user privacy; attacks on implantable cardiac devices, which may result in battery depletion or incorrect pacing and electric shock, endangering the lives of patients; attacks on children's smartwatches, which can expose their location data and personal information, creating various safety hazards; and attacks on the Controller Area Network (CAN) bus of vehicles, which can manipulate speed and direction, leading to potential casualties. A smart home hacking investigation revealed that, in a week, smart home devices, such as televisions, thermostats, smart kettles, and security systems, were attacked by cybercriminals and unknown entities more than 12,000 times (Wang et al., 2022). An Intrusion Detection System (IDS) is a network security system that identifies abnormal intrusion traffic through network data analysis and intercepts the intrusion traffic, significantly reducing the success rate of network attacks. In recent years, numerous researchers have proposed methods for safeguarding network security through the use of intrusion detection techniques. For example, Bakalos et al. (2019) utilized visual surveillance, channel state information (CSI) from Wi-Fi signals, and ICS sensor data from the utility to detect intrusions in water infrastructure. Notably, their study encompassed not only network intrusions but also

* Corresponding author.
E-mail addresses: wangzhendong@jxust.edu.cn (Z. Wang), 6120210180@mail.jxust.edu.cn (J. Li), yangshuxin@jxust.edu.cn (S. Yang), luoxiao@jxust.edu.cn (X. Luo), lidahai@jxust.edu.cn (D. Li).
[1] https://orcid.org/0009-0008-7095-9096.

https://doi.org/10.1016/j.eswa.2023.122045
Received 20 March 2023; Received in revised form 23 August 2023; Accepted 4 October 2023
Available online 10 October 2023

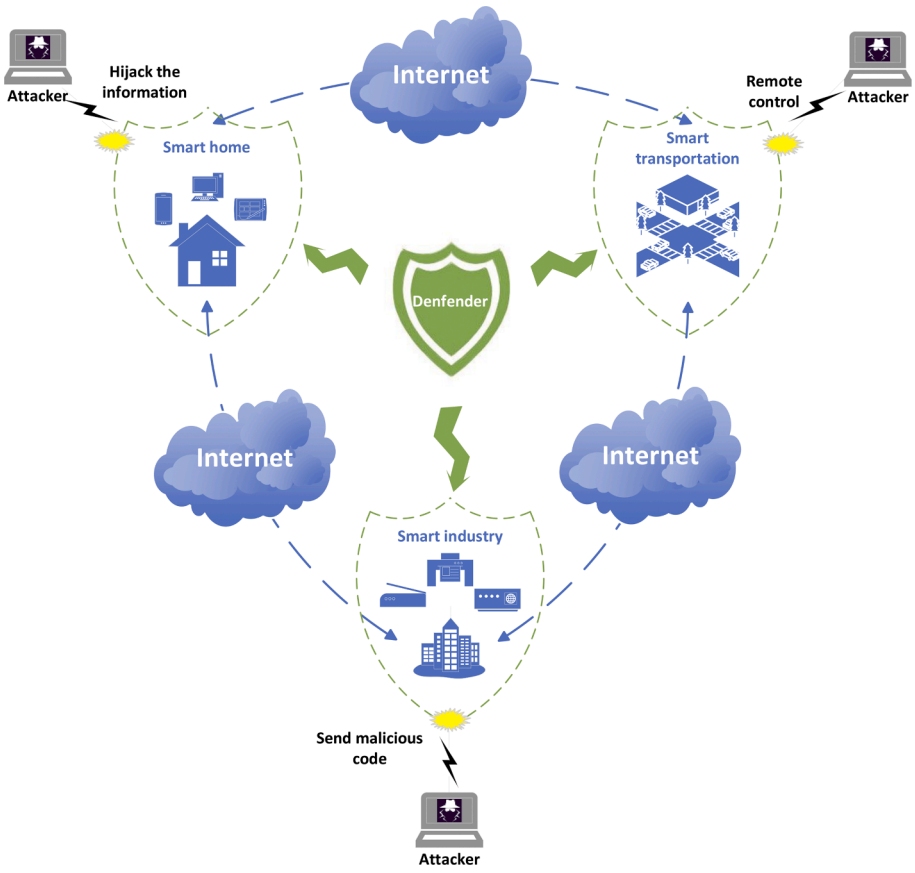


Fig. 1. Attack-Defense architecture of IoT.

physical intrusions by humans. Their research comprehensively considered the potential threats to water infrastructure, and thus has significant implications for related research endeavors. Unlike other network security systems, IDS is an active network security protection technology that detects network intrusion data in near-real-time. It can activate defense measures when a threat occurs, thereby preventing the further expansion of the scope of security threats and reducing the damage caused by network security threats to a minimum. Hence, IDS is a critical technology in ensuring the security of IoT networks. Fig. 1 illustrates the IoT attack-defense architecture.

While traditional intrusion detection systems have been highly successful in securing the networks of conventional devices, they may not be suitable for IoT networks due to the limited computing power of IoT devices. Typically, IoT devices are microcomputing devices with restricted computing resources (Su et al., 2018). Additionally, they have limited communication and storage capabilities, and their IoT network resources are also highly constrained. Therefore, it is not feasible to deploy traditional IDSs directly in IoT networks (Roy et al., 2022). In 2021, Panagiotis et al. (Panagiotis et al., 2021) published a literature review that provided a systematic overview of intrusion behaviors and detection methods. They underscored that network intrusion could cause severe damage to critical infrastructure, and emphasized the necessity of developing lightweight intrusion detection systems that can operate on resource-constrained devices.

Deep Learning algorithms are often used in traditional intrusion detection systems due to their high accuracy and ability to generalize well. However, DL-based intrusion detection methods require significant computational resources, making large-scale DL models unsuitable for real-time attack detection in IoT networks. Therefore, it is important to investigate methods for compressing intrusion detection models without sacrificing detection performance. Various techniques for compressing deep learning models exist, including Parameter Quantization (Gong et al., 2014), Weight Pruning (Han, Mao, & Dally, 2015), and Knowledge Distillation (KD) (Hinton, Vinyals, & Dean, 2015). Among these techniques, KD is particularly attractive due to its simple implementation, significant model compression effects, and minimal loss of model performance after compression. KD has been widely used, particularly in the fields of Computer Vision (CV) (Zhu et al., 2021) and Natural Language Processing (NLP) (Tang et al., 2019), with great success. KD involves using a pre-trained large neural network model as a teacher model to guide the training of a smaller neural network model (student model). By minimizing the distillation loss function, knowledge is transferred from the teacher model to the student model, resulting in improved performance of the student model. This enables the smaller student model to achieve performance levels similar to those of the larger teacher model.

Knowledge distillation techniques facilitate the compression of larger teacher models into smaller student models while maintaining their performance. This technology is highly suitable for IoT intrusion detection, which has limited computational resources. Existing knowledge distillation methods use distillation loss to minimize the gap between teacher and student models. To reduce the complexity of this technique, Xu et al. (2020) proposed a method called BERT-of-Theseus, which compresses the BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) model through module replacement. This method uses low-complexity modules as substitutes to mimic the original ones, resulting in reduced computational costs. Specifically, this paper employs a nine-layer Vision Transformer (ViT) (Dosovitskiy et al., 2020) as the large model (teacher model) and a three-layer PoolFormer (Yu et al., 2022) as the small model (student model). Subsequently, an improved version of BERT-of-Theseus is utilized to enable the student model to possess the detection performance of the teacher model. We refer to the aforementioned intrusion detection method as an IoT intrusion detection method based on BERT-of-Theseus,

Vision Transformer, and PoolFormer, named BT-TPF. Through experimental evaluation on the CIC-IDS2017 and Ton-IoT datasets, we select the optimal loss function and network structure. Finally, on the test dataset, our model demonstrates improved training efficiency and accuracy compared to traditional methods.

The main contributions of this paper to the intrusion detection area include the following:

1) Based on the characteristics of network intrusion behavior data, Vision Transformer and PoolFormer are used for intrusion detection, and their effectiveness in improving the accuracy of network intrusion detection is proven.
2) A lightweight model for ViT Guided Distill PoolFormer is proposed based on KD's new model, BERT-of-Theseus. It reduces the model size while maintaining the effectiveness of network intrusion detection, and it is suitable for IoT devices with limited resources.
3) The gradient propagation process of BERT-of-Theseus is theoretically analyzed, and the knowledge distillation process of BERT-of-Theseus is optimized according to the analysis results. This optimization significantly accelerates the knowledge distillation process.
4) A dimensionality reduction method based on a Siamese network is proposed, utilizing the advantages of deep metric learning to encode input features. It reduces the feature similarity of samples within the same category and increases the feature similarity of samples from different categories.

The remaining sections of this paper are organized as follows. Section 2 provides a review of related work on intrusion detection in IoT networks. Section 3 presents the proposed framework for knowledge distillation models based on BERT-of-Theseus for IoT intrusion detection. Section 4 describes the dataset used in this study and its preprocessing steps. Section 5 introduces the experimental setup, including the evaluation metrics, hyperparameters, and baseline model. Section 6 presents the experimental results and evaluation of the proposed BT-TPF model on the CIC-IDS2017 and TON_IoT datasets. Finally, Section 7 summarizes the conclusions of the paper.

2. Related works

This section focuses on the intrusion detection models applied in the IoT and introduces some intrusion detection models based on Transformer in recent years.

2.1. IoT intrusion detection methods

Based on the detection mechanism used in the system, Zarpelão et al. (2017) classified IDS in IoT into four categories: signature-based IDS, anomaly-based IDS, specification-based IDS, and hybrid-based IDS. Among these categories, signature-based and anomaly-based methods are the most widely used. Signature-based IDSs are highly accurate and effective in detecting known threats. These IDSs identify system or network behavior as an intrusion when it matches the attack signature stored in their internal database (Liao et al., 2013). Kasinathan et al. (2013) proposed a signature-based IDS for 6LoWPAN-based IoT networks to identify Denial-of-Service (DoS) attacks in IoT networks. The architecture integrates an IDS into the network framework developed within the EU FP7 project ebbits. The IDS sends alerts to the DoS protection manager, which analyzes additional information such as the channel interference rate and packet drop rate to confirm the attack, thereby reducing the false positive rate. However, how this work keeps updating the attack signatures is unknown. Sheikh et al. (2019) devised a signature-based intrusion detection system and proposed a pattern-matching algorithm to efficiently detect known network attack vectors, such as smurf and ipseep. While signature-based IDSs are effective in detecting known network attacks, they are ineffective in detecting novel attacks or variations of known attacks, for which the corresponding matching characteristics are still unknown. Additionally, since a new signature must be generated for every attack variant, the system's detection speed decreases as the number of rules increases. Consequently, the overall ability of the signature engine to detect these changes will be affected.

Anomaly-based intrusion detection systems (IDS) compare the current system activity to a normal behavior profile and generate alerts when deviations from normal behavior exceed a threshold. This approach can be effective in detecting new attacks. There are several approaches to anomaly-based IDS, including Machine Learning (ML) methods. Safaldin, Otair, and Abualigah (2021) proposed an improved SVM-based binary grey wolf optimizer (GWOSVM-IDS) for IDS. The GWOSVM-IDS employs three, five, and seven wolves to find the optimal number of wolves and selects the most important features to improve classification accuracy and reduce the false alarm rate of intrusion detection. Kumar, Gupta, and Tripathi (2021) proposed a distributed ensemble IDS based on IoT fog computing, which uses mutual information to select the optimal subset of features and employs K-Nearest Neighbor (KNN), XGBoost, and Gaussian Naive Bayes as first-level individual learners. Finally, the prediction results obtained from the first-level individual learners are classified by random forest. De Souza, Westphall, and Machado (2022) proposed an approach for IoT intrusion detection based on feature selection and ensemble. They used the Recursive Feature Elimination (RFE) wrapper technique with Extra Trees (ET) to select the best features for the detection process. In the first detection stage, ET is used to classify whether the input is benign. In the second stage, anomalous traffic that cannot be identified as benign is multi-classified by an ensemble approach consisting of ET, Random Forest, and Deep Neural Network. It is well known that most ML-based methods rely on feature engineering. Among these methods, wrapper-based feature selection utilizes the performance of the learning algorithm to evaluate the features' merits. However, when the learning algorithm is changed, feature selection must be performed again, which can lead to high computational complexity and longer execution time, particularly for larger-scale datasets. Furthermore, the classifier must be trained and tested for each evaluation of the subset, which contributes to the algorithm's overall complexity. In dealing with multiple classification problems, SVM, KNN, and other classifier models typically face issues of high computational complexity and long detection time. Hence, traditional machine learning-based techniques are inadequate for comprehensively tackling the challenge of large-scale intrusion detection in real-world IoT application environments.

Jayalaxmi et al. (2022) devised a Deep Learning model using a cascaded forward-back propagation neural network (CFBPNN) for detecting industrial network traffic. The CFBPNN model offers dynamic recurrent features of forward and backward connectivity, enabling the reconnection of multiple network layers and tracking of variable impacts at each layer. Asadi (2022) employed cooperative game theory with three methods: Long Short-Term Memory (LSTM), Autoencoder, and Support Vector Machine (SVM). The experimental results have demonstrated that the Autoencoder exhibits excellent overall performance and remarkable speed in detecting botnet traffic. Shahin et al. (2022)) presented a Convolutional Neural Network with Long Short-Term Memory (CNN-LSTM) and a Long Short-Term Memory Fully Convolutional Network (LSTM-FCN), augmenting the LSTM with CNN and FCN. The results show that the DL algorithm can accurately detect and classify over 99.9% of the attacks in two instances of the three datasets. Mothukuri et al. (2021) proposed an anomaly detection method based on Federated Learning (FL), which uses data on distributed devices to actively identify intrusions in IoT networks. The method utilizes federated training wheels on a Gated Ryclic Unit (GRU) model and keeps the data on local IoT devices intact by sharing only the learned weights with the FL's central server. Experimental results demonstrated that the method outperformed the centralized machine learning (non-FL) version in safeguarding user data privacy and providing the best accuracy in attack detection. Zhang et al. (2020)

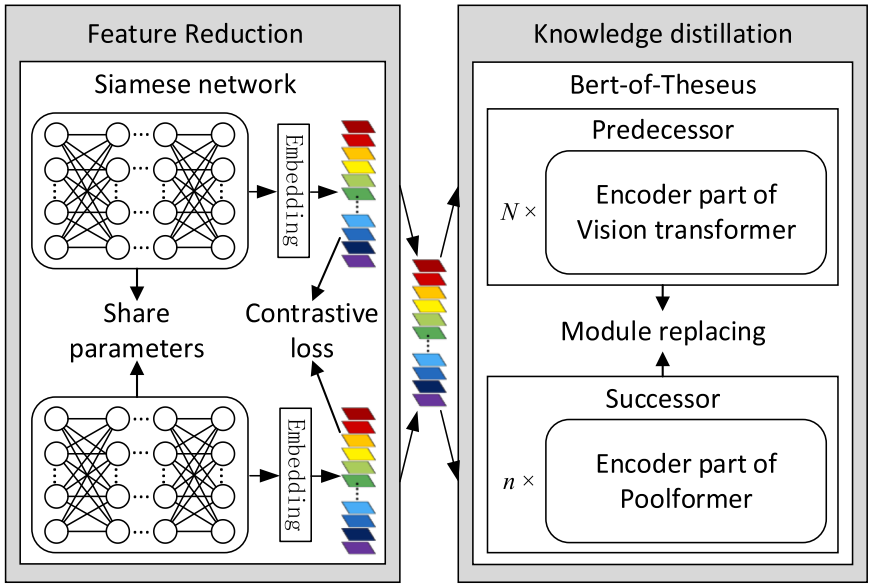


Fig. 2. Overall architecture of the BT-TPF model.

devised SGM-CNN, a flow-based intrusion detection model that utilizes SMOTE and GMM-based undersampling clustering techniques. By combining imbalanced class processing with convolutional neural networks, the approach effectively enhances the accuracy of IoT attack detection.

Prior studies have demonstrated the successful usage of DL techniques to identify anomalous network traffic in IoT. Nonetheless, DL models often have a large scale and a substantial number of parameters, making it challenging to deploy them to every node in the IoT network due to the nodes' limited power, computing capability, communication capability, and storage capacity. Thus, it is necessary to develop a lightweight network intrusion detection model with a small model size and fewer training parameters that is suitable for IoT intrusion detection.

2.2. Intrusion detection method based on the Transformer network model

The Transformer (Vaswani et al., 2017) was originally developed for use in Natural Language Processing (NLP) tasks. Unlike traditional network structures such as RNN and CNN, the Transformer relies solely on the Attention mechanism for machine translation tasks. Since its introduction, Transformer-based models have made significant breakthroughs in NLP, with BERT (Devlin et al., 2018) proposed by Devlin et al. and GPT-3 (Brown et al., 2020) proposed by Brown et al. being notable examples. Compared to RNN-based temporal neural networks, the Transformer is significantly different in terms of training methodology. While RNN training is iterative and serial, the Transformer's training is parallel. This parallel training approach allows the Transformer to train all features simultaneously, resulting in improved computational efficiency. The underlying idea behind the Transformer is to define neural network layers entirely based on the attention mechanism. At each layer, a new hidden representation is generated for each location by computing the attention weights for all locations in the input data.

In their study, Wu et al. (2022) proposed a robust intrusion detection system (IDS) based on the Transformer architecture, called RTIDS. This method employs positional embedding technology to correlate sequence information between features and utilizes a variable stacked encoder-decoder neural network for feature extraction. Furthermore, it applies a self-attention mechanism for network traffic classification. To prevent overfitting, the authors designed a variant Transformer with an additional masked multi-headed self-attention sublayer in the decoder stack. Experimental results show that RTIDS outperforms the mainstream classical algorithms used in other IDSs. However, as intrusion detection systems require rapid responses, the speed of the Transformer algorithm needs to be improved.

Research on Transformer-based architectures in the field of Computer Vision has yielded impressive results, as demonstrated by ViT (Dosovitskiy et al., 2020). Ho et al. (2022) applied a windowing and overlapping mechanism to convert varying input sizes to a standard-size RGB image for classification. The Vision Transformer (ViT) classifier was then employed to classify the resultant images. Experimental results showed that binary classification accuracy on the CIC-IDS2017 dataset reached 98.5%. However, the presence of data imbalance in the dataset resulted in inaccurate multi-classification results.

With the advancement of technology, machine learning and deep learning methods have made significant progress in intrusion detection. This paper proposes a BERT-of-Theseus-based model compression method for IoT intrusion detection, which compresses a ViT-based 9-layer network intrusion detection model and applies it to the IoT benchmark datasets CIC-IDS2017 and TON_IoT. This method considers the limited power of nodes, fewer computing resources, poor communication environment, limited storage capacity, and limited model generalization ability in the IoT network. The model's performance is evaluated using various metrics such as Accuracy, Recall, Precision, F1-Score, model size, and number of parameters, making the evaluation more thorough and reasonable. This method provides an effective solution for intrusion detection in IoT networks with limited resources.

3. Methodology

This section is devoted to the KD-based intrusion detection framework BT-TPF, as proposed in this study. Fig. 2 presents the comprehensive structure of the BT-TPF model. To begin, the pre-processed data is subjected to processing by a Siamese network for the purpose of generating its corresponding embedding. Subsequently, the reshape layer is employed to transform the embedding into an image representation. Thereafter, the improved BERT-of-Theseus approach, which is based on module replacement, is utilized to conduct knowledge distillation on the pre-trained ViT model and Poolformer model. This leads to the creation of the Poolformer model that exhibits improved intrusion detection accuracy while having fewer parameters.

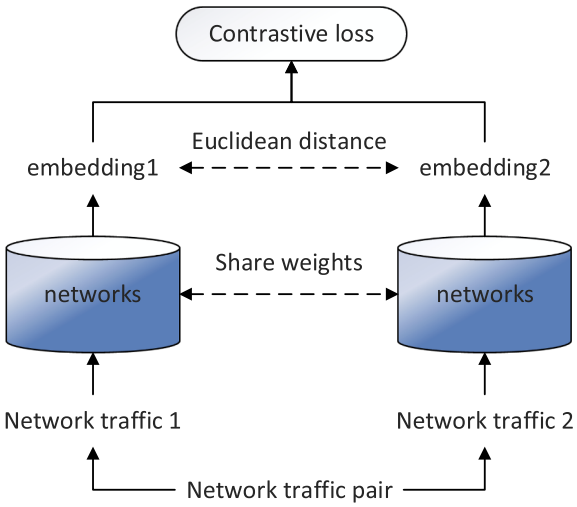


Fig. 3. The architecture of the Siamese network.

3.1. Feature dimensionality reduction method based on the Siamese network

The Siamese network is a type of comparative learning that utilizes a neural network to map input data to a projection space. This projection space employs a distance function, such as Euclidean distance or Cosine similarity, as a similarity measure to evaluate the resulting projection output. During the training phase, the neural network parameters are optimized to minimize the similarity value between a pair of samples from the same category and maximize the similarity value between samples from different categories. Due to the ability of the Siamese network to effectively capture the similarity between data, increasing the similarity of feature representations for the same type of data and decreasing it for different types of data, we employ the Siamese network for feature extraction from network data in this paper. The resulting neural network can map samples of the same category to similar positions in the projection space while simultaneously positioning samples of different categories far apart in the projection space, thus achieving effective dimensionality reduction.

The Siamese network architecture utilized in this paper consists of three layers of Multilayer Perceptron (MLP) with shared parameters. The network processes pairs of inputs, generating distinct encoded outputs for each. As shown in Fig. 3, the Siamese network transforms the left and right input pairs into new vectors within the projection space and calculates the distance between the two vectors to determine their similarity. In this study, the Euclidean distance is utilized as the similarity metric, as illustrated in Equation (1).

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \tag{1}$$

The objective of training a Siamese network is to minimize the distance between two samples of the same category in the projection space and maximize the distance between two samples of different categories in the projection space. Hence, to achieve this objective, the Contrastive Loss function is utilized as the loss function of the Siamese network in this study, as depicted in Eq. (2).

$$L = \frac{1}{2N}\sum_{n=1}^{N} yd^2 + (1-y)max(margin - d, 0)^2 \tag{2}$$

where d denotes the Euclidean distance between two samples, and y represents the label indicating whether the samples belong to the same category or not. Specifically, $y = 1$ when the samples belong to the same category, and $y = 0$ when they belong to different categories. The variable margin denotes the set threshold, which is a positive number representing the minimum distance to be achieved by dissimilar samples. The magnitude of margin determines the degree of attention that the network pays to distinguish different types of samples. The larger margin is, the more attention the network pays to distinguish different types of samples. When $y = 1$ (i.e., the two samples belong to the same category), the loss function is given by $\sum d^2$, implying that if the Euclidean distance between samples in the feature space is large, the current model is ineffective, and the loss value needs to be increased. When $y = 0$ (i.e., the samples belong to different categories), the loss function is given by $\sum max(margin - d, 0)^2$, meaning that the smaller the Euclidean distance d in the feature space between non-similar samples, the larger the loss value. The loss value is 0 when the Euclidean distance d is greater than the threshold m. This Contrastive Loss function effectively captures the similarity of paired samples and can be used to train the model for feature dimensionality reduction.

3.2. Intrusion detection model based on BERT-of-Theseus

In the conventional knowledge distillation method, a range of mechanisms are employed to ensure that the small model's performance approximates that of the large model upon completion of training. This is achieved by designing various loss functions to minimize the discrepancies between the output of the small model and the output of the original large model. Commonly used loss functions include KL-divergence (Hinton, Vinyals, & Dean, 2015) and Mean-Squared Loss (Tung & Mori, 2019). Additionally, more intricate methods involve designing loss functions to assess the differences between intermediate layer outputs of the two models (Sun et al., 2019).

Since the introduction of the Transformer, researchers have proposed numerous deep learning models based on this architecture. Among these models, BERT has garnered significant attention. BERT, short for Bidirectional Encoder Representations from Transformers, is a highly performant pre-trained language model based on the Transformer architecture. While BERT exhibits impressive performance, its large model size and high computational cost pose challenges. To address this, Xu et al. (2020) proposed the Bert-of-Theseus method, which compresses the BERT model while preserving its performance. This approach effectively reduces the model size while maintaining the desirable characteristics of the original BERT model.

Contrasting with traditional knowledge distillation methods, BERT-of-Theseus does not require complex loss functions to evaluate differences between large and small models. Instead, BERT-of-Theseus utilizes a module replacement mechanism to enhance the performance of the smaller model by making it more similar to the larger model. This study first trains a high-accuracy large model, Predecessor, and subsequently leverages the BERT-of-Theseus method to distill the knowledge of the Predecessor into a smaller model, Successor, that achieves comparable performance. Finally, the small model is used for intrusion detection in IoT devices that have limited computing resources. The subsequent section outlines the methodology for achieving a high-precision intrusion detection model for IoT via BERT-of-Theseus.

As per the BERT-of-Theseus framework, both Predecessor and Successor are composed of n modules. We denote the Predecessor and Successor as P and S, respectively, which can be represented using Eqs. (3) and (4).

$$P = \{prd_1, prd_2, \cdots, prd_i, \cdots, prd_n\} \tag{3}$$

$$S = \{scc_1, scc_2, \cdots, scc_i, \cdots, scc_n\} \tag{4}$$

Where the variables prd_i and scc_i respectively denote the i-th module of the Predecessor and the i-th module of the Successor. In the knowledge distillation process using BERT-of-Theseus, the Predecessor's parameters are frozen, and the Predecessor and Successor are merged into a comprehensive model (Mix). Subsequently, Mix undergoes training to execute knowledge distillation.

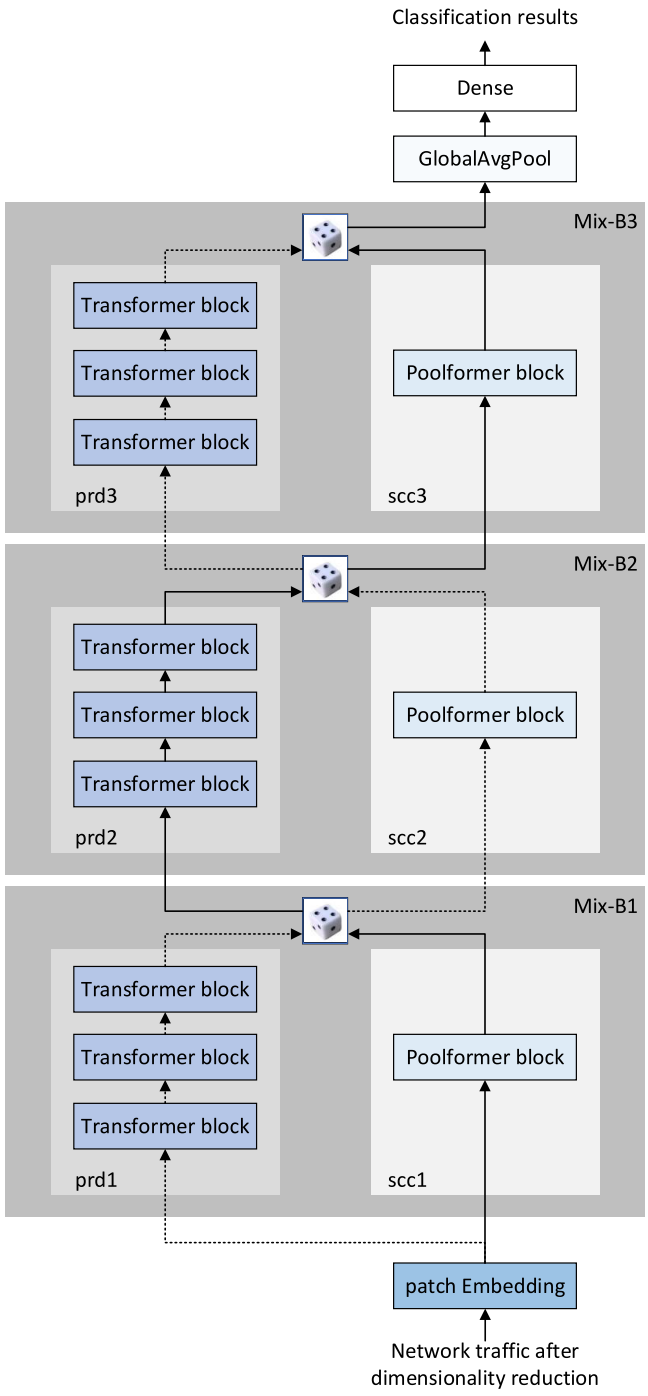


Fig. 4. The knowledge distillation framework of BERT-of-Theseus.

As illustrated in Fig. 4, the Mix model comprises multiple submodules, with submodule $Mix-B_i$ being composed of prd_i in P and scc_i in S. The outputs of prd_i and scc_i are respectively denoted as y_i^p and y_i^s. Consequently, the output y_{i+1} of the i + 1th submodule $Mix-B_{i+1}$ in Mix can be mathematically represented through Eqs. (5) and (6).

$$r_{i+1} \sim Bernoulli(p) \tag{5}$$

$$y_{i+1} = r_{i+1} * prd_i(y_i) + (1 - r_{i+1}) * scc_i(y_i) \tag{6}$$

where r_{i+1} represents an independent random variable conforming to the Bernoulli distribution, $r_{i+1} \in \{0, 1\}$, p denotes the replacement rate, and the * symbol denotes element-wise multiplication. Each module of the Predecessor and its corresponding module of the Successor function cooperatively during the training process.

In the replacement training process, this paper utilizes the Mean Squared Error (MSE) loss function, as shown in Eq. (7).

$$L = mean\left[(y - label)^2\right] \tag{7}$$

where y represents the output of the Mix model, label denotes the corresponding label, and $mean(*)$ signifies the mean value function.

3.3. Distillation optimization

During backpropagation, the weights of all Predecessor modules are frozen to facilitate the computation of gradients between the Predecessor and Successor modules, thereby enabling deeper interactions. Upon convergence of the module interaction replacement training, the Successor modules are collected and recombined to form a Successor model. As the size of each Successor module is smaller than the corresponding Predecessor module, the previous model P is effectively compressed into a smaller model S. Subsequently, the model S is fine-tuned by optimizing the loss function in Eq. (7). During the fine-tuning phase, all Successor modules are fully integrated into the training task to minimize the discrepancy between the training and inference processes.

To expedite the distillation process, this study conducts an in-depth analysis of the gradient in BERT-of-Theseus and devises optimization strategies accordingly. Specifically, we scrutinize the gradient propagation method in BERT-of-Theseus, enhance the selection probability of the last layer, augment the gradient value within a manageable range, and accelerate the neural network learning process. The detailed procedure of our analysis is presented below, along with the optimized outcomes.

Given that the parameters in the Predecessor module are frozen, it follows that solely the parameters in the Successor module undergo updating during the knowledge distillation process in BERT-of-Theseus. When the output of the i-th submodule in Mix is the output of the Mix model, the gradient of the i-th module in Successor can be represented using Eq. (8), as per the relationship presented in Eq. (6).

$$\partial \mathrm{L}(\mathrm{y_{i+1}}, \mathrm{y_{label}})/\partial\theta_s = (1 - \mathrm{r_{i+1}}) \bullet \partial \mathrm{L}(\mathrm{y_{i+1}}, \mathrm{y_{label}})/\partial \mathrm{scc_i}(\mathrm{y_i}) \bullet \partial \mathrm{scc_i}(\mathrm{y_i})/\partial\theta_s \tag{8}$$

In the case where the loss function is defined as the MSE function, $\mathrm{L}(\mathrm{y_{i+1}}, \mathrm{y_{label}})$ can be mathematically formulated using Eq. (9).

$$\begin{aligned} L(y_{i+1}, y_{label}) &= (y_{i+1} - y_{label})^2 \\ &= [r_{i+1} \bullet prd_i(y_i) + (1 - r_{i+1}) \bullet scc_i(y_i) - y_{label}]^2 \end{aligned} \tag{9}$$

By substituting Eq. (9) into Eq. (8), the gradient of the i-th module in Successor can be reformulated as presented in Equation (10).

$$\begin{aligned} \partial L(y_{i+1}, y_{label})/\partial\theta_i^s = 2[r_{i+1} \bullet prd_i(y_i) + (1 - r_{i+1}) \bullet scc_i(y_i) - y_{label}](1 \\ - r_{i+1})\partial scc_i(y_i)/\partial\theta_i^s \end{aligned} \tag{10}$$

Eq. (10) can be further simplified, leading to the expression represented by Eq. (11).

$$\begin{aligned} \partial L(y_{i+1}, y_{label})/\partial\theta_i^s = 2\big[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) \\ + y_{label})r_{i+1} + scc_i(y_i) - y_{label}\big]\partial scc_i(y_i)/\partial\theta_i^s \end{aligned} \tag{11}$$

In order to expedite the training convergence, the adaptive adjustment of the gradient during the training process becomes essential. Eq. (11) illustrates that the magnitude of the gradient increases as the absolute value of $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}]$ increases. By appropriately amplifying the gradient, the neural network convergence can be accelerated. Moreover, as per the findings of literature (Xu et al., 2020), the phenomenon of

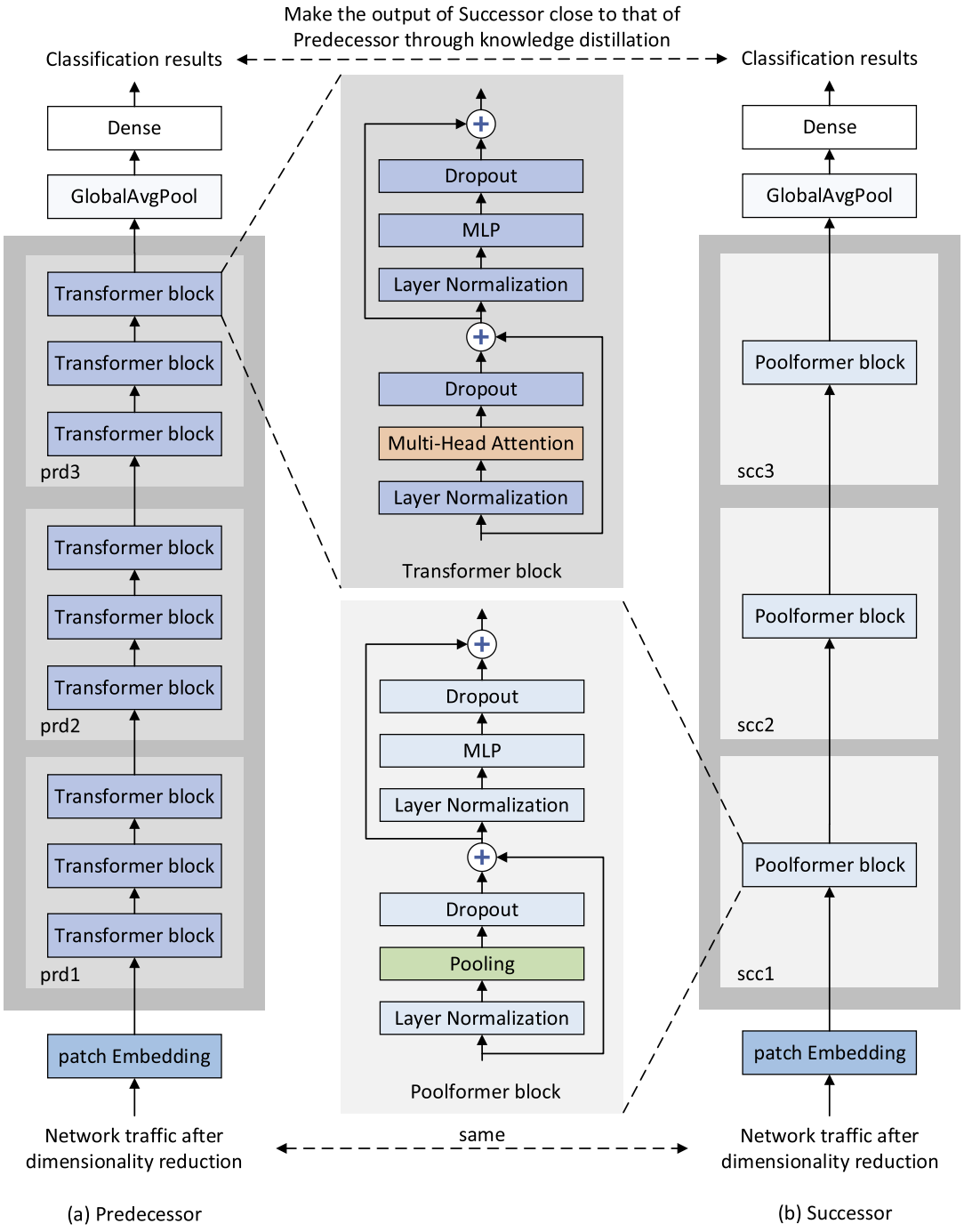


Fig. 5. The architecture of the Predecessor and Successor.

vanishing or exploding gradient is unlikely to occur when $r_{i+1} \in (0,1)$. By analyzing Eq. (11), it can be observed that $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}]$ can only attain its maximum absolute value at three positions, $r_{i+1} = 0$, $r_{i+1} = 1$, or $r_{i+1} = -(prd_i(y_i) - 2scc_i(y_i) + y_{label})/2(scc_i(y_i) - prd_i(y_i))$.

Below is a detailed analysis of the three aforementioned cases:

A. When $r_{i+1} = 0$, $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}] = scc_i(y_i) - y_{label}$.

B. When $r_{i+1} = 1$, $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}] = 0$. The gradient value is at its minimum, and the vanishing gradient phenomenon is expected to occur. Therefore, when $r_{i+1} = 1$, the absolute value of $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}]$ is the smallest. Therefore, this case should be excluded.

C. When $r_{i+1} = -(prd_i(y_i) - 2scc_i(y_i) + y_{label})/2(scc_i(y_i) - prd_i(y_i))$, $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}]$ obtains extreme values Γ.

The aforementioned analysis demonstrates that the maximum absolute value of $[(scc_i(y_i) - prd_i(y_i))r_{i+1}{}^2 + (prd_i(y_i) - 2scc_i(y_i) + y_{label})r_{i+1} + scc_i(y_i) - y_{label}]$ can be attained by comparing the magnitudes of $|scc_i(y_i) - y_{label}|$ and $|\Gamma|$. Consequently, the value of r_{i+1} can be expressed by Eq. (12).

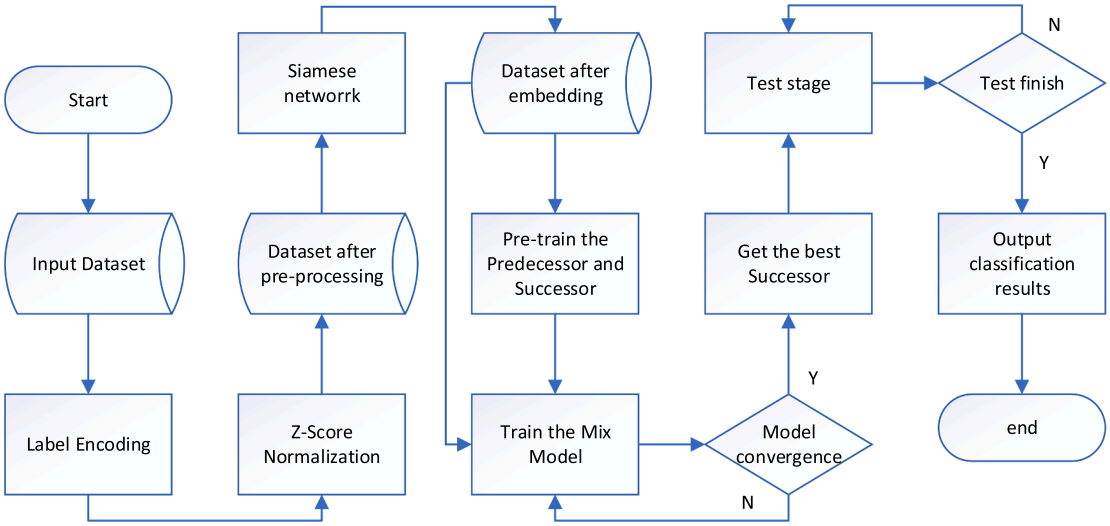


Fig. 6. Flow chart of the BT-TPF model.

$$r_{i+1} = \begin{cases} 0 | scc_i(y_i) - y_{label} | \geq |\Gamma| \\ -(prd_i(y_i) - 2scc_i(y_i) + y_{label})/2(scc_i(y_i) - prd_i(y_i)) | scc_i(y_i) - y_{label} | < |\Gamma| \end{cases} \tag{12}$$

In combination with Equation (12), Equation (11) can be rewritten as Equation (13).

$$\partial L(y_{i+1}, y_{label})/\partial\theta_i^s = \begin{cases} 2(scc_i(y_i) - y_{label})\partial scc_i(y_i)/\partial\theta_i^s | scc_i(y_i) - y_{label} | \geq |\Gamma| \\ 2\Gamma\partial scc_i(y_i)/\partial\theta_i^s | scc_i(y_i) - y_{label} | < |\Gamma| \end{cases} \tag{13}$$

This paper utilizes the probability selection approach outlined in the literature (Xu et al., 2020) for the remaining layers, without any further modification.

3.4. Predecessor and Successor

The process of model compression through knowledge distillation involves transferring knowledge from a larger model with superior performance to a smaller one. As a result, it becomes essential to construct a high-performing intrusion detection model with a large-scale architecture. The Transformer's Multi-Head Attention mechanism has been shown to effectively capture feature correlations, thus significantly enhancing intrusion detection accuracy. Notably, current Transformer-based intrusion detection approaches have achieved outstanding outcomes in network intrusion detection (Wu et al., 2022; Ho et al., 2022). Consequently, this paper proposes the Predecessor model, which leverages the Transformer architecture to construct a large-scale intrusion detection model with superior detection accuracy.

The Transformer architecture typically requires a token sequence as input, which is represented as a two-dimensional matrix. The Vision Transformer (ViT) (Dosovitskiy et al., 2020) implements a patch embedding approach that leverages a convolutional layer to divide the input feature map $X \in \mathbb{R}^{H\times W\times C}$ into patches. These patches are then utilized to construct a two-dimensional matrix $X_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, as depicted in Eq. (14).

$$X_p = [x_1; x_2; \cdots; x_N], N = HW/P^2 \tag{14}$$

where, (H, W) represents the dimensions of the initial feature map, C denotes the number of channels, (P, P) refers to the size of each patch, and N represents the number of patches created. The number of generated patches is equivalent to the length of the valid input token sequence for the Transformer architecture.

As the Transformer architecture lacks the inherent ability to discern positional relationships between input elements, positional encoding is crucial for this task. The encoding process is accomplished via Sine and Cosine functions with varying frequencies. Specifically, odd positions are encoded with Cosine functions, while even positions are encoded with Sine functions. These operations are mathematically represented by Eqs. (15)–(17):

$$PE(pos, 2i) = sin\left(pos/1000^{2i/d_{model}}\right) \tag{15}$$

$$PE(pos, 2i+1) = cos\left(pos/1000^{2i/d_{model}}\right) \tag{16}$$

$$z_0 = X_p + PE \tag{17}$$

where, pos denotes the position, i signifies the dimension, and $PE(*)$ represents the positional encoding for each dimension, which is associated with the curve.

The Transformer architecture is essentially an Encoder-Decoder structure, wherein the Transformer Encoder comprises a series of coding blocks that are stacked L times. Although the coding blocks possess identical structures, they do not share weights. Fig. 5(a) illustrates each coding block, which consists of two sublayers. The first sublayer encompasses Layer Normalization (LN) (Ba, Kiros, & Hinton, 2016), Multi-Head Self-Attention (MHA), and Residual Connections (He et al., 2016), while the second sublayer consists of Layer Normalization (LN), MLP, and Residual Connections.

In the Transformer Encoder, the input data undergoes a preliminary phase wherein it is subjected to the Multi-Head Self-Attention layer, resulting in a feature vector z that is weighted accordingly. This process can be represented by Eq. (18):

$$z_l^{'} = D(MHA(LN(z_{l-1}))) + z_{l-1}, l = 1, \cdots, L \tag{18}$$

The obtained value of Z is subsequently transmitted to the subsequent layer of the encoder, as demonstrated in Eq. (19).

Table 1
Network traffic distribution of CIC-IDS2017.

Attack category	Description	Train	Test
Benign	Normal network traffic.	329,762	109,921
GoldenEye	The attacker utilizes a large number of HTTP POST requests to occupy server resources, resulting in the server being unable to respond to legitimate requests.	7720	2573
Hulk	The attacker sends a large number of GET requests, each of which carries a significant amount of random data, in order to consume server resources.	172,593	57,531
Slowhttptest	The attacker sends a large number of slow requests, which maintain long-lasting connections and slowly transmit data, thereby occupying server resources.	4124	1375
Slowloris	The attacker sends a large number of half-open connections, which remain in a prolonged state without fully establishing the connection, thereby occupying server resources.	4347	1449
Total	–	518,546	172,849

$$z_l = D\left(MLP(LN(z_l^{'}))\right) + z_l^{'}, l = 1, \cdots, L \tag{19}$$

The coding blocks from the Transformer Encoder serve as the foundational components of the Predecessor module in this study. The Predecessor model's 9-layer network is partitioned into three modules, with each module comprising three Transformer coding blocks.

Poolformer (Yu et al., 2022) is a variation of the Transformer architecture, as illustrated in Fig. 5(b). In the Poolformer coding block, the Multi-Head self-Attention layer is substituted with a pooling layer that contains no trainable parameters. This layer serves as the token mixer, while the remaining layers of Poolformer remain consistent with Transformer. Notably, the computational complexity of Self-Attention is quadratic in the number of tokens, whereas the complexity of pooling is linearly proportional to the sequence length. Additionally, the Poolformer coding block is uncomplicated and lacks learnable parameters. Thus, this block serves as the fundamental component of the Successor module in this study, offering a lightweight design advantage to the Successor model. The Successor model's 3-layer network is partitioned into three modules, with each module comprising one Poolformer coding block.

3.5. The intrusion detection steps of the BT-TPF model

The BT-TPF model for intrusion detection consists of the following detailed steps:

Step 1: Data pre-processing: This step involves the numerical processing of character-based data and data normalization for intrusion detection datasets. Further information regarding this step can be found in Section 4.2.

Step 2: Dimensionality reduction using the Siamese network: The intrusion detection dataset is fed into the Siamese network to reduce its dimensionality.

Step 3: Pre-training of the Predecessor and Successor models: The Predecessor and Successor models are pre-trained before knowledge distillation.

Step 4: Knowledge distillation using the improved BERT-of-Theseus: In this step, the weights of the pre-trained Predecessor model are fixed, and the corresponding module of the Successor model is randomly replaced with the Predecessor module. After sufficient training, the entire Successor model is separated and fine-tuned until the verification dataset indicators no longer rise.

Step 5: Testing for the Successor model: The pre-processed test dataset is fed into the Successor model to obtain the classification results of each network traffic.

Table 2
Network traffic distribution of TON_IoT.

Attack category	Description	Train	Test
Backdoor	The attacker, by implanting or exploiting vulnerabilities in systems, applications, or network devices, clandestinely creates a hidden backdoor, thereby bypassing normal security mechanisms and remotely accessing and controlling the target system.	15,000	5000
DDos	The attacker, by remotely controlling zombie computers or devices, simultaneously sends a large number of requests to the target server, causing it to be unable to process legitimate user requests and resulting in service unavailability.	15,000	5000
Dos	The attacker, by sending a large number of requests to the target server or occupying its resources, causes it to be unable to process legitimate user requests, resulting in service unavailability.	15,000	5000
Injection	The attacker exploits application vulnerabilities to inject malicious code into the application, thereby executing unauthorized operations.	15,000	5000
Mitm	The attacker inserts their own node between the two ends of communication in order to steal communication data or to impersonate and tamper with communication content.	782	261
Normal	Normal network traffic.	225,000	75,000
Password	The attacker attempts to obtain user passwords through various means in order to illegally access user accounts or systems.	15,000	5000
Ransomware	A type of malware attack where the attacker encrypts files on the user's computer and demands payment of a ransom to decrypt the files.	15,000	5000
Scanning	The attacker scans open ports and services in the target network or system to gather information about the target, including possible vulnerabilities and weaknesses.	15,000	5000
Xss	A common web application security vulnerability where attackers inject malicious script code into trusted web pages, causing it to execute in the user's browser.	15,000	5000
Total	–	345,782	115,261

The overall flow chart of the BT-TPF model is illustrated in Fig. 6.

4. Data analysis and pre-processing

In order to evaluate the effectiveness of the BT-TFP model in detecting IoT intrusions, this study performs intrusion detection experiments on the IoT benchmark datasets CIC-IDS2017 (Sharafaldin, Lashkari, & Ghorbani, 2018) and TON_IoT (Moustafa, 2019; Moustafa et al, 2020a; 2020b; Alsaedi et al., 2020; Moustafa, 2021a; Moustafa, 2021b; Ashraf et al., 2021; Booij et al., 2021).

4.1. Dataset description

The CIC-IDS2017 dataset, which was released by the Canadian Institute of CyberSecurity (CIC) in 2017, is composed of benign traffic as well as various attack types commonly observed in real-world scenarios, such as DoS attacks. The dataset comprises around 3 million network traffic data, stored in eight CSV files, with each data consisting of 78 different features and their respective labels. To reasonably reduce the model training duration and to fulfill the multi-classification requirement, we select the Wednesday-workingHours.csv file for the experiments. Due to the small number of Heartbleed attacks present in this

dataset subset (only 11), we exclude these samples from our analysis. Consequently, the dataset used in this study includes a total of 691,395 instances belonging to five categories, with 75% of the data being allocated to the training set and the remaining 25% to the test set. This partitioning approach ensures that the system is trained on a sufficiently large sample size and tested on an independent subset of the data, thereby mitigating overfitting and improving generalization. The network traffic distribution of the CIC-IDS2017 dataset is presented in Table 1.

The TON_IoT datasets have been obtained from a comprehensive and practical network, which was created at the Cyber Range and IoT Labs of the School of Engineering and Information Technology (SEIT), UNSW Canberra @ the Australian Defense Force Academy (ADFA). These datasets, present in the Train_Test_datasets folder, comprise CSV-formatted data samples that are appropriate for assessing the precision and efficacy of Machine Learning algorithms in new cybersecurity applications. Each dataset instance includes 43 features, along with their respective labels. The features can be classified into four categories (namely, connection-related, statistical, violation-related, and user-related attributes) based on the service profile principles. We select the Train_Test_Network dataset as a suitable dataset for training and testing the intrusion detection system in the IoT environment. To ensure optimal performance, the dataset will be split into a training set and a test set, with 75% of the data allocated to the former and the remaining 25% to the latter. The network traffic distribution of the TON_IoT dataset is presented in Table 2.

4.2. Data pre-processing

The training and testing data must conform to the input format of the model, so the experimental dataset needs to be pre-processed, which is an important step in intrusion detection. The pre-processing stages are as follows:

Raw data inevitably contain redundant and incomplete samples, which may negatively affect classification accuracy. To overcome this problem, these records are removed from the dataset in this paper. For example, the CIC-IDS2017 dataset has data with “NaN”, “Infinity”, and null values, so they are removed from the dataset.

Character feature encoding encompasses two principal techniques: One-Hot Encoding and Label Encoding. The former increases the dimensionality of the dataset and may impair the performance and efficiency of the model. In contrast, Label Encoding maps each symbolic feature category to a corresponding integer. For instance, in the TON_IoT dataset, there are three attributes in the proto feature, namely, tcp, udp and icmp, and then these three attributes can be processed as 0, 1, and 2. This paper uses Label Encoding to convert symbolic features into numeric features. Additionally, Nan, dash, and infinity values are replaced with 0, and boolean features are assigned a value of 1 when true and 0 when false.

The intrusion detection dataset exhibits substantial variability in the range of values for each of its features, which has an impact on both model training and convergence speed. To address this, it is necessary to perform feature scaling on the dataset. One effective technique for this purpose is Z-Score Normalization, which utilizes the mean and standard deviation of the original data to normalize it. As a result, the processed data conform to the standard normal distribution, characterized by a mean of 0 and a standard deviation of 1. The transformation function is expressed as follows:

$$x_{new} = x - \mu/\sigma \tag{20}$$

where μ and σ represent the mean and standard deviation of the sample data, respectively. Moreover, normalized data derived through this method retain valuable information pertaining to outliers, rendering the algorithm less sensitive to them. In contrast, this cannot be ensured by utilizing Min-Max normalization.

Finally, A Siamese network was constructed, wherein the model with shared parameters comprises a three-layer MLP consisting of an input layer, a hidden layer, and an output layer. Following numerical processing and normalization, the network traffic was fed into the model, and features of specified dimensions were extracted through encoding. The Siamese network facilitates the amplification of the similarity between samples of the same category, while simultaneously reducing the similarity between samples of different categories, following feature extraction. Next, a reshaping operation was performed to transform the network traffic into a 6x6x1 feature map.

5. Experimental setup

The hardware environment used in this experiment is configured as an AMD Ryzen 7 5800H processor, 16 GB RAM, and a single NVIDIA RTX 3060 GPU used for computing acceleration. The operating system is Windows 11.

5.1. Evaluation metrics

Due to the intricacy of the network intrusion detection dataset and the evident issue of data imbalance, the evaluation metrics for the intrusion detection model will encompass Accuracy, Recall, Precision, and F1-Score. The metrics for assessing the model’s performance are defined as follows:

Accuracy: The ratio of the number of samples correctly classified by the model to the total number of samples, as shown in Equation (21):

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{21}$$

Recall: The ratio of the number of positive samples correctly identified by the model to all positive samples in the total number of samples, as shown in Eq. (22):

$$Recall = TP/(FN + TP) \tag{22}$$

Precision: The ratio of true positive samples to the number of samples identified by the model as positive, as shown in Eq. (23):

$$Precision = TP/(TP + FP) \tag{23}$$

F1-Score: F1-Score can be regarded as the harmonic mean of model precision and recall, as shown in Eq. (24):

$$F1 = 2TP/(2TP + FP + FN) \tag{24}$$

where True Positive (TP) represents the number of positive samples correctly predicted as positive; True Negative (TN) denotes the number of negative samples correctly predicted as negative; False Positive (FP) signifies the number of negative samples incorrectly predicted as positive; and False Negative (FN) signifies the number of positive samples incorrectly predicted as negative. All these values can be obtained from the confusion matrix.

Through a series of experiments comparing the aforementioned evaluation metrics, the performance of the intrusion detection model can be comprehensively examined, and crucially, the superiority of the proposed model can be demonstrated. Additionally, the model size and the number of parameters are also used as evaluation metrics to quantify the computational resources necessary for the intrusion detection model.

5.2. Hyperparameter setting

The manual selection of hyperparameter values can significantly impact a model’s classification performance. Thus, conducting a series of experiments is crucial for determining the optimal hyperparameter values in a model. The Siamese network comprises a three-layer MLP consisting of an input layer, a hidden layer, and an output layer. The number of neurons in the hidden layer is set to 5, and the threshold

Table 3
Baseline of CIC-IDS2017.

Model	Accuracy	Precision	Recall	F1-Score	Parameters	ModelSize
Predecessor	0.9967	0.9967	0.9967	0.9967	13,440	52.5 KB
Successor	0.9803	0.9766	0.9803	0.9747	918	3.6 KB

Table 4
Baseline of TON_IoT.

Model	Accuracy	Precision	Recall	F1-Score	Parameters	ModelSize
Predecessor	0.9958	0.9958	0.9958	0.9958	13,310	52 KB
Successor	0.9742	0.9770	0.9742	0.9717	788	3.1 KB

Table 5
The evaluation metric results for CIC-IDS2017 without using the Siamese network.

Model	Accuracy	Precision	Recall	F1-Score	Parameters	ModelSize
Predecessor	0.9855	0.9855	0.9855	0.9855	12,829	50.1 KB
Successor	0.9274	0.9115	0.9274	0.9157	307	1.2 KB

Table 6
The evaluation metric results for TON_IoT without using the Siamese network.

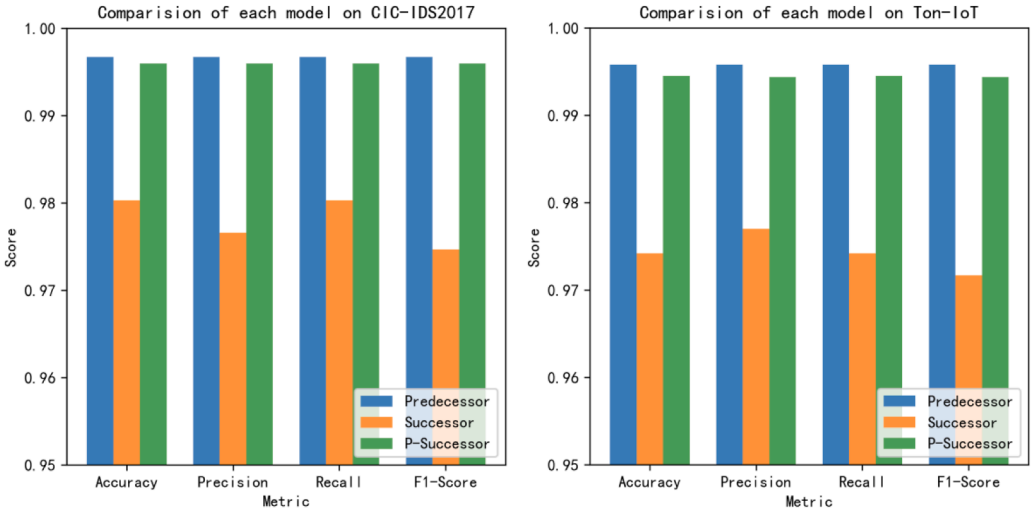
Model	Accuracy	Precision	Recall	F1-Score	Parameters	ModelSize
Predecessor	0.9789	0.9798	0.9789	0.9790	12,874	50.3 KB
Successor	0.9410	0.9389	0.9410	0.9347	352	1.4 KB

margin in Contrastive Loss is set to 1. In patch embedding, the convolution step is set to 2, the convolution kernel size is set to 2, and the number of convolution kernels is set to 8. In the Predecessor, the number of heads in the Multi-Head Attention layer is set to 2, and the number of neurons in the MLP block middle layer is set to 4 times the length of the token sequence (patches). In Successor, to minimize the size and number of parameters of the Successor model, the number of MLP block middle layer neurons is set to 1. Therefore, the successor model is suitable for IoT devices with limited resources. Finally, a pooling layer without trainable parameters is connected to a fully connected layer in the classify module to output the classification results.

zIn the next experiments, we use Adam (Kingma, & Ba., 2014) as the gradient optimizer and Tanh as the activation function. The batch size is set to 1024, and the initial learning rate is set to 0.0001. The epoch in the pre-training phase is set to 50, and the epoch in the hybrid training phase based on module replacement is set to 250. During the fine-tuning phase, the Successor is trained until the validation dataset indicators no longer improve.

Based on the aforementioned hyperparameters, we input the pre-processed CIC-IDS2017 and TON_IoT datasets into the Predecessor and Successor models, respectively, for pre-training. The experimental results obtained from pre-training serve as the baseline.

As depicted in Table 3 and Table 4, the number of parameters in the Predecessor model is 14 times greater than that in the Successor model, and the Predecessor outperforms the Successor in all evaluation indicators. A comparison between the baseline and our model will be presented later in this paper.



(a) CIC-IDS2017 (b) TON_IoT

Fig. 7. Performance comparison of each model.

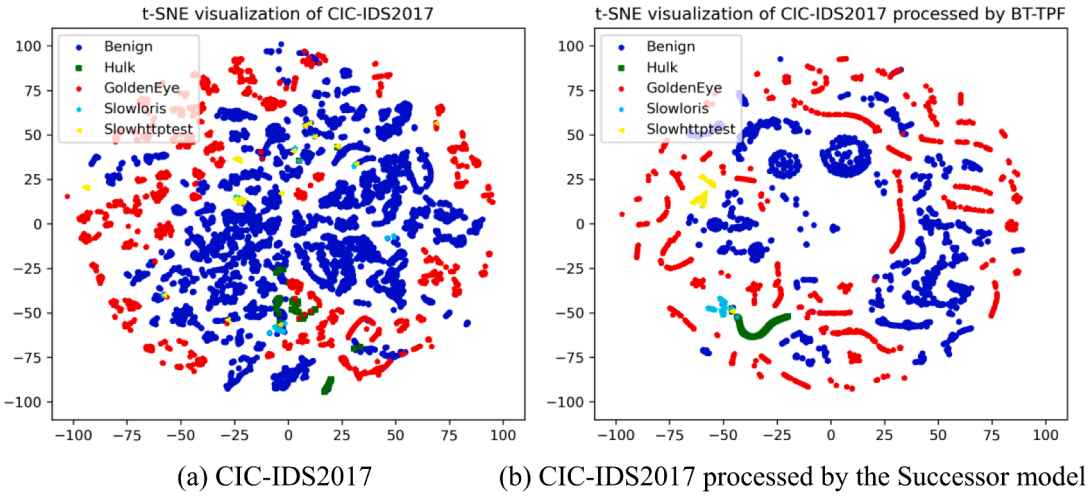


Fig. 8. t-SNE visualization of CIC-IDS2017.

6. Results and discussion

To demonstrate the effectiveness of our proposed BT-TPF model, we will conduct comprehensive experiments on the IoT intrusion detection datasets CIC-IDS2017 and TON_IoT. Specifically, our evaluation will focus on two aspects: the feature dimensionality reduction method based on the Siamese network and the Predecessor Guided Distill Successor approach based on module replacement. Multiple evaluation metrics described in Section 5.1 are used to measure the superiority of the BT-TPF model with traditional deep learning approaches and the recent state-of-the-art models.

6.1. Classification results of the CIC-IDS2017 and TON_IoT datasets

In this study, we employ the Siamese network to reduce the dimensionality of input features, with the aim of enhancing intrusion detection capabilities. Specifically, we apply this approach to two original datasets, CIC-IDS2017 and TON_IoT, encoding their feature dimensions from 78 to 36 and from 43 to 36, respectively. Our method effectively reduces the features and data size required for intrusion detection. To evaluate the efficacy of our feature dimensionality reduction approach, we conducted a series of ablation experiments. Results from Table 5 and Table 6 demonstrate that the accuracy of the Predecessor model without feature coding was relatively low, at 98.55% and 97.89% for the CIC-IDS2017 and TON_IoT datasets, respectively, while the accuracy of the Successor model without feature coding was 92.74% and 94.10%, respectively. In contrast, Table 3 and Table 4 indicate that after incorporating the Siamese network for feature encoding and dimensionality reduction, the accuracy of the Predecessor model increased significantly to 99.67% and 99.58% for the CIC-IDS2017 and TON_IoT datasets, respectively, while the accuracy of the Successor model improved to 98.03% and 97.42%, respectively. These findings suggest that our proposed method is able to efficiently extract the most important features, thereby reducing computational complexity and yielding a lightweight model that is well-suited for IoT networks with resource constraints.

To establish the superiority of the BT-TPF model over the baseline model, we trained the former on the IoT intrusion detection benchmark dataset and compared its performance with that of the latter. The findings of this comparative analysis are presented in Fig. 7. Specifically, both the Predecessor and the Successor were trained using a pre-processed IoT intrusion detection dataset. Notably, the Predecessor model featured 14 times more parameters than the Successor and BT-TPF models, which accounts for its superior performance on the evaluation metrics. As evidenced by the detection results of the CIC-IDS2017 dataset depicted in Fig. 7(a), the BT-TPF model performed similarly to the Predecessor model across all evaluation metrics, with a difference of only 0.07% in their overall performance. In contrast, the BT-TPF model exhibited superior performance compared to the Successor model, with accuracy and recall that were 1.57% higher, as well as precision and F1-Score that were 1.94% and 2.13% higher, respectively. Moreover, the detection results of the TON_IoT dataset depicted in Fig. 7(b)

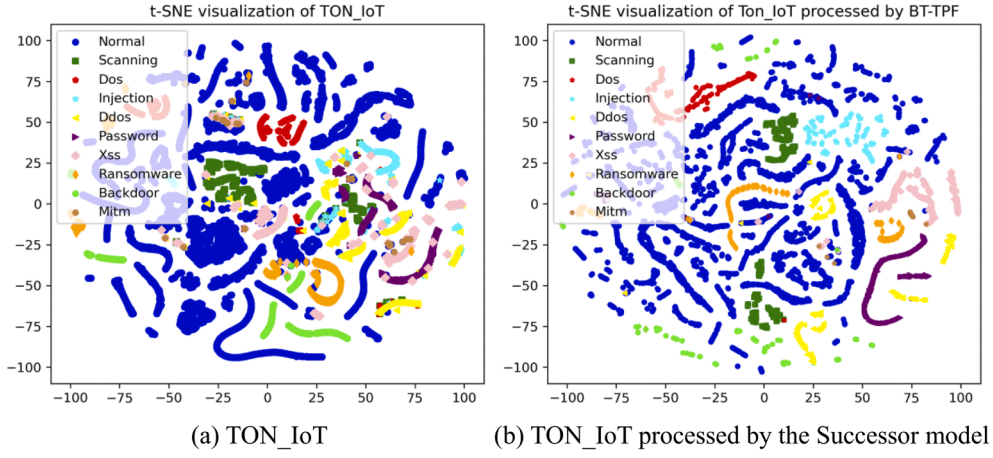


Fig. 9. t-SNE visualization of TON_IoT.

Table 7
Multi-classification results of the CIC-IDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score	Parameters	ModelSize
DNN	0.9789	0.9792	0.9789	0.9789	1096	4.3 KB
CNN	0.9762	0.9759	0.9762	0.9753	1333	5.2 KB
LSTM	0.9732	0.9713	0.9732	0.9679	1837	7.2 KB
BT-TPF	0.9960	0.9960	0.9960	0.9960	918	3.6 KB

demonstrated that the BT-TPF model performed similarly to the Predecessor model across all evaluation metrics. The accuracy of the BT-TPF model was only 0.13% lower than that of the Predecessor model, but 2.03% higher than that of the Successor model.

This is primarily due to the following reasons: (1) In knowledge distillation training, a module replacement strategy is employed to ensure that the output of the Successor network closely resembles that of the Predecessor network. Through this process, the Successor network acquires classification knowledge from the Predecessor network, which enhances its ability to identify attacks on minority samples. (2) The Successor network is capable of fitting the samples in advance, but it may get stuck in a local optimal region during the later stages of iteration. Employing the concept of pre-training and fine-tuning, the pre-trained Predecessor model will be co-trained with the Successor model, thereby leading the Successor model towards a better direction and increasing its possibility of escaping from the local optimum. (3) As time advances during the distillation training, the Successor network has a higher probability of being involved in the training process, thus rendering it more robust even in the absence of the Predecessor network. Consequently, it can be inferred that the Predecessor Guided Distill Successor approach based on module replacement reduces the number of model parameters while simultaneously improving the performance of the Successor model.

To provide an intuitive demonstration of the efficacy of our multi-classification experiment, we randomly selected 20,000 network traffic samples and subjected them to processing with the BT-TPF model. We then used the t-SNE algorithm (Van der Maaten, & Hinton, 2008) to visualize the resulting outputs. t-SNE (t-Distributed Stochastic Neighbor Embedding) is a nonlinear dimensionality reduction algorithm used to map high-dimensional data into a low-dimensional space for visualization or further analysis. The key idea of the t-SNE algorithm is to preserve the similarity relationships between data points in the high-dimensional space when constructing the similarity relationships between data points in the low-dimensional space. This is achieved by computing the similarity probabilities between high-dimensional data points and the similarity probabilities between low-dimensional data points. For comparison, we also passed the same 20,000 samples through t-SNE without any processing. The visualization results for both the CIC-IDS2017 and TON_IoT datasets are presented in Fig. 8 and Fig. 9, respectively. As depicted in Fig. 8(a), the distribution of data for each category in the CIC-IDS2017 dataset is relatively proximate, with Hulk attacks, Slowloris attacks, and Slowhttptest attacks appearing scattered among the Benign and GoldenEye clusters. By comparison, the BT-TPF-processed data shown in Fig. 8(b) effectively distinguish the various categories of attack data, resulting in a clearer demarcation between Benign, Hulk, and GoldenEye clusters. In Fig. 9(a), the TON_IoT dataset has more attack categories than the CIC-IDS2017 dataset, resulting in a more intricate distribution of various attack types. Notably, Xss attacks and Ddos attacks exhibit significant overlap with other types, leading to potential misclassification into other categories. However, as illustrated in Fig. 9(b), BT-TPF processing produces substantial improvements over the unprocessed data, with Xss attacks and DDoS attacks forming clusters at the periphery of other data clusters. The visualized images from both the original and BT-TPF-processed datasets reveal an aggregation tendency for different categories of network traffic in the low-dimensional space. In summary, our BT-TPF model demonstrates high accuracy in identifying different categories of abnormal network traffic.

6.2. Comparison with traditional deep learning models and advanced intrusion detection models

This paper utilizes conventional deep learning algorithms, namely Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Long Short-Term Memory Network (LSTM), to perform comparative experiments and assess the efficacy of the proposed BT-TPF model. In order to ensure the validity of the experiments, DNN, CNN, and LSTM models that possess similar structures and parameter numbers to the BT-TPF model are utilized. Table 7 and Table 8 exhibit the experimental outcomes of multi-classification using both the traditional DL models and the BT-TPF model on the CIC-IDS2017 and TON_IoT datasets, respectively. Table 7 demonstrates that on the CIC-IDS2017 dataset, the BT-TPF model's multi-classification detection accuracy is 1.71%, 1.98%, and 2.28% higher than that of the DNN, CNN, and LSTM models, respectively. Furthermore, Table 8 illustrates that the BT-TPF model outperforms all other models in multi-classification detection on the TON_IoT dataset, with all metrics exhibiting better performance than the other models. The key factor contributing to this outcome is the pre-trained Predecessor model's ability to help the Successor model avoid local optima and develop a superior feature representation. Overall, in intrusion detection of the CIC-IDS2017 and TON_IoT datasets, our proposed BT-TPF model not only has a smaller parameter count and model size than DNN, CNN, and RNN but also demonstrates superior multi-classification detection efficacy than traditional deep learning models.

The confusion matrix is a graphical tool that employs heatmaps, color differences, and brightness to represent variations in data. It has the ability to combine the instances in a dataset according to the actual value and the predicted result, thus enabling the extraction of multiple evaluation metrics for a model. Fig. 10 and Fig. 11 illustrate the confusion matrices of all the models on the CIC-IDS2017 and TON_IoT datasets, respectively. Notably, it can be inferred from Fig. 10(d) and - Fig. 11(d) that the BT-TPF model outperforms the other models in terms of classification results. On the CIC-IDS2017 dataset, the LSTM model misclassified 73% of Slowloris as Benign and 53% of Hulk as GoldenEye, whereas the CNN model misclassified 49% of Slowloris as Benign. Similarly, on the TON_IoT dataset, the DNN model misclassified 54% of Mitm and 38% of Xss as Normal, while the RNN model misclassified 50% of Mitm as Normal, and the CNN model misclassified 57% of Mitm as Normal and 37% of Mitm as Xss. The traditional deep learning models

Table 8
Multi-classification results of the TON_IoT dataset.

Model	Accuracy	Precision	Recall	F1-Score	Parameters	ModelSize
DNN	0.9378	0.9338	0.9378	0.9347	966	3.8 KB
CNN	0.9601	0.9582	0.9601	0.9586	1418	5.5 KB
LSTM	0.9653	0.9641	0.9653	0.9644	1818	7.1 KB
BT-TPF	0.9945	0.9945	0.9945	0.9944	788	3.1 KB

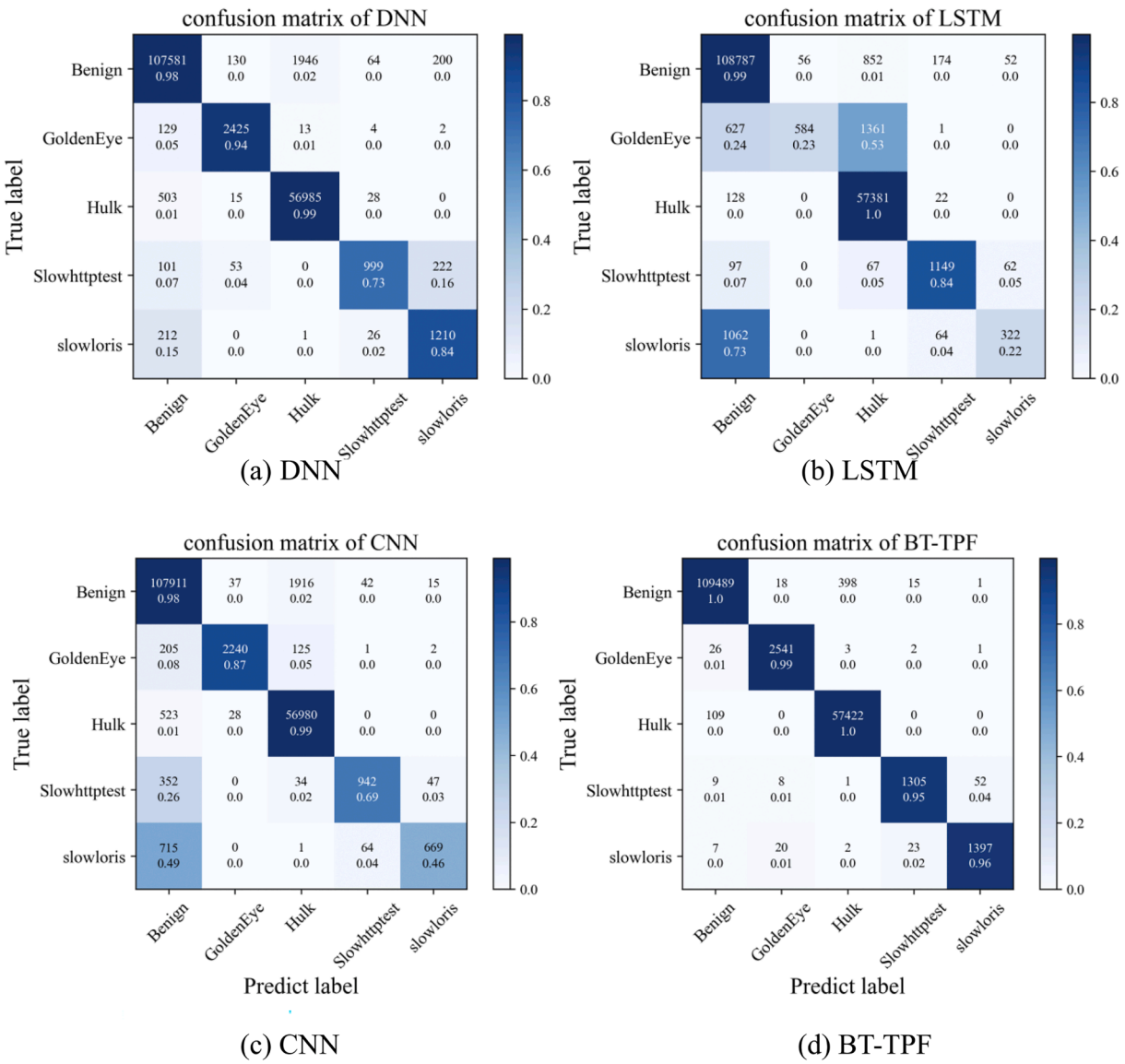


Fig. 10. Multi-classification confusion matrix for each model on CIC-IDS2017.

faced challenges in detecting Slowloris and Mitm attacks due to a limited number of samples, leading to numerous false positives and false negatives. In contrast, the proposed BT-TPF model achieved an accuracy of 96% and 87% in detecting these two few-sample attacks, highlighting its effectiveness in intrusion detection.

To further demonstrate the superiority of the BT-TPF model, this paper compares it with the latest advanced models on the CIC-IDS2017 and Ton-IoT datasets. Table 9 and Table 10 below present the experimental results, where "-" indicates that the performance metrics are not reported in these papers. The BT-TPF model is a lightweight model, so its detection results are a trade-off between the number of parameters and detection efficiency. As shown in Table 9, the BT-TPF model is only lower than the SVM-GAC model in terms of accuracy and recall, but its accuracy is respectively improved by 0.16%, 2.31%, 0.36%, and 1.56% compared to the KD-TCNN, Bi-RNN, B-AnoGLA, and Multi-strategy + PCA models. In terms of F1-Score, the BT-TPF model achieves a result of 99.60%, which is respectively higher than the TempoCode-IoT, KD-TCNN, Bi-RNN, B-AnoGLA, and Multi-strategy + PCA models by 0.69%, 1.53%, 0.14%, 2.3%, 0.88%, and 1.16%, indicating that the proposed model can effectively detect and classify network intrusion traffic. As shown in Table 10, the accuracy of the BT-TPF model is only 0.06% lower than that of the STFA-HDLID model, but it is respectively higher than the RF, Chi2-XGBoost, AE-DT, XGB tuned MLP, and DeepAK-IoT models by 1.38%, 1.25%, 1.22%, 1.33%, and 8.88%. In terms of precision and F1-Score, the BT-TPF model outperforms other models. In conclusion, the above models can obtain better results for various performance indicators in the intrusion detection process of the Ton-IoT dataset.

6.3. *Evaluating the lightweight performance of the BT-TPF model*

The objective of this study is not only to achieve excellent classification performance but also to prioritize lightweight performance for better deployment in IoT devices. As IoT devices typically have limited computing and communication resources, we compared our proposed algorithm model with other well-known lightweight deep learning models that can be deployed on resource-constrained devices to determine if our model is lightweight. Specifically, we evaluated the lightweight performance of our proposed model by comparing the parameter count and computational cost (FLOPs, MAdds) of different models. The parameter count represents the number of parameters that need to be learned in the model, and FLOPs and MAdds represent the number of floating-point operations and multiply–add operations that the model needs to perform during the inference stage, serving as indicators to

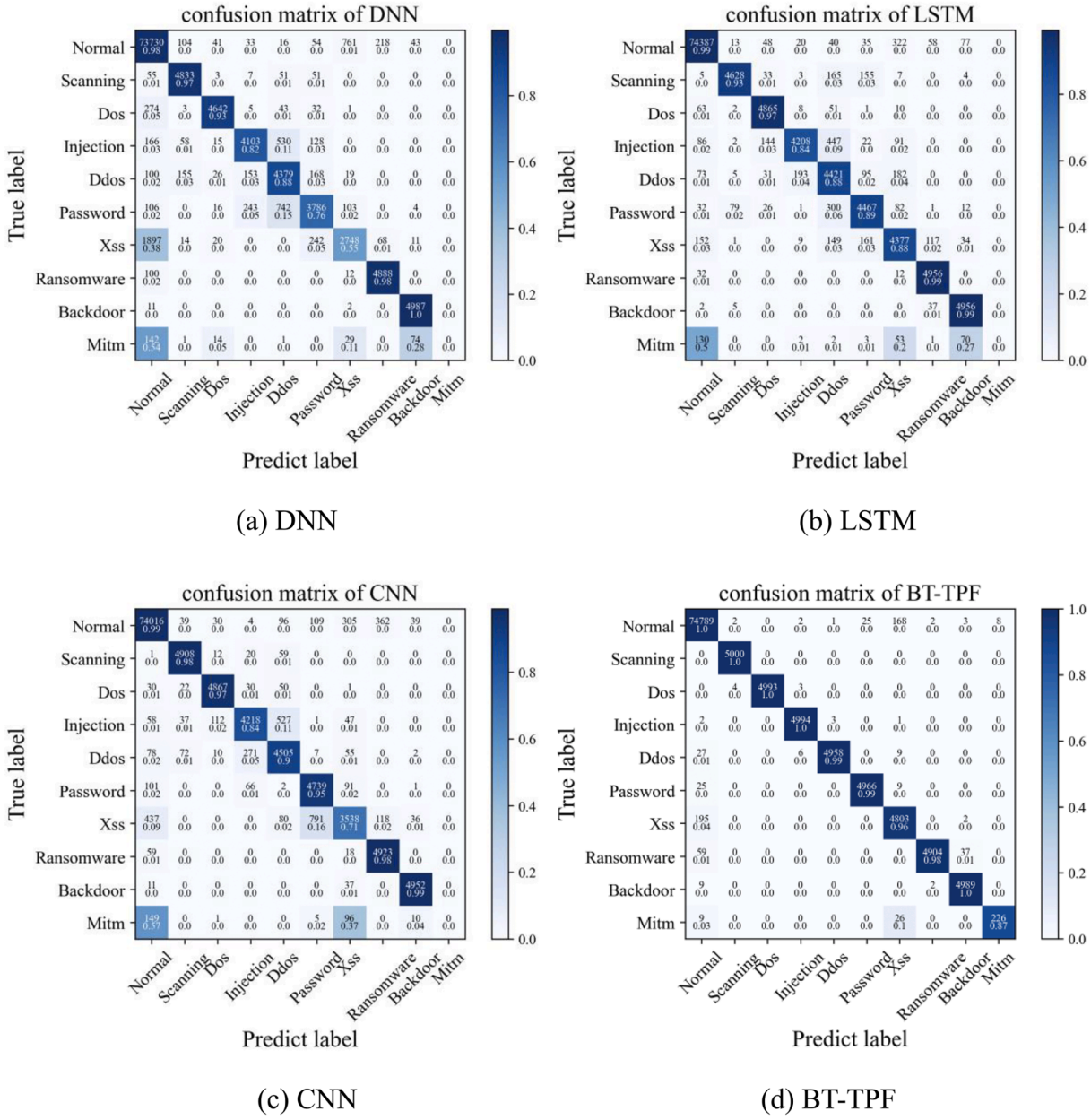


Fig. 11. Multi-classification confusion matrix for each model on TON_IoT.

Table 9
Comparison results of the CIC-IDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
SVM-GAC (Shukla, 2021)	0.9993	0.9873	0.9961	0.9891
TempoCode-IoT (Siddiqui & Boukerche, 2021)	–	0.9842	0.9772	0.9807
KD-TCNN (Wang et al., 2022)	0.9944	0.9948	0.9947	0.9946
Bi-RNN (Gautam et al., 2022)	0.9729	0.9732	0.9729	0.9730
AnoGLA (Ding & Li, 2022)	0.9924	0.9850	0.9862	0.9872
Multi-strategy + PCA (Li et al., 2022)	0.9804	0.9906	0.9785	0.9844
BT-TPF (our model)	0.9960	0.9960	0.9960	0.9960

Table 10
Comparison results of the TON_IoT dataset.

Model	Accuracy	Precision	Recall	F1-Score
RF (Booij et al., 2021)	0.9807	–	–	0.9726
Chi2-XGBoost (Gad, Nashat, & Barkat, 2021)	0.982	0.982	0.982	0.982
AE-DT (Sarhan et al., 2022)	0.9823	–	–	0.99
STFA-HDLID (Escorcia-Gutierrez et al.,2023)	0.9951	–	–	0.9753
XGB tuned MLP (Kilincer et al.,2023)	0.9812	0.9815	0.9812	0.9812
DeepAK-IoT (Ding, Abdel-Basset, & Mohamed, 2023)	0.9057	0.8959	–	0.8887
BT-TPF	0.9945	0.9945	0.9945	0.9944

evaluate the computational complexity of the model. Additionally, since the CIC-IDS2017 and TON_IoT datasets have different numbers of data features, we evaluated the lightweight performance of the model on both datasets. From Table 11, it is evident that compared to other well-known lightweight models, our model has lower computational cost, smaller parameter count, and smaller Model Size. This indicates that our model is sufficiently lightweight.

Table 11
Comparison results with well-known lightweight models.

Model	CIC-IDS2017				TON_IoT			
	Params	Flops	MAdds	Model Size	Params	Flops	MAdds	Model Size
mobilenet_v2 (Sandler et al., 2018)	3.5 × 10^6	3.6 × 10^6	7.2 × 10^6	13.4 MB	3.5 × 10^6	3.6 × 10^6	7 × 10^6	13.4 MB
mobilenet_v3_small (Howard et al., 2019)	2.5 × 10^6	2.6 × 10^6	5.1 × 10^6	9.7 MB	2.5 × 10^6	2.6 × 10^6	5.1 × 10^6	9.7 MB
efficientnet_b0 (Tan & Le, 2019)	5.3 × 10^6	5.4 × 10^6	10.7 × 10^6	20.2 MB	5.3 × 10^6	5.4 × 10^6	10.6 × 10^6	20.2 MB
mnasnet0_5 (Tan et al., 2019)	2.2 × 10^6	2.3 × 10^6	4.5 × 10^6	8.5 MB	2.2 × 10^6	2.2 × 10^6	4.4 × 10^6	8.5 MB
mnasnet0_75 (Tan et al., 2019)	3.2 × 10^6	3.3 × 10^6	6.5 × 10^6	12.1 MB	3.2 × 10^6	3.2 × 10^6	6.4 × 10^6	12.1 MB
mnasnet1_0 (Tan et al., 2019)	4.4 × 10^6	4.5 × 10^6	8.9 × 10^6	16.7 MB	4.4 × 10^6	4.4 × 10^6	8.8 × 10^6	16.7 MB
mnasnet1_3 (Tan et al., 2019)	6.3 × 10^6	6.4 × 10^6	12.8 × 10^6	24 MB	6.3 × 10^6	6.3 × 10^6	12.6 × 10^6	24 MB
shufflenet_v2_x0_5 (Ma et al., 2018)	1.4 × 10^6	1.4 × 10^6	2.8 × 10^6	5.2 MB	1.4 × 10^6	1.4 × 10^6	2.7 × 10^6	5.2 MB
shufflenet_v2_x1_0 (Ma et al., 2018)	2.3 × 10^6	2.3 × 10^6	4.6 × 10^6	9.7 MB	2.3 × 10^6	2.3 × 10^6	4.6 × 10^6	9.7 MB
shufflenet_v2_x1_5 (Ma et al., 2018)	3.5 × 10^6	3.5 × 10^6	7 × 10^6	13.4 MB	3.5 × 10^6	3.5 × 10^6	7 × 10^6	13.4 MB
shufflenet_v2_x2_0 (Ma et al., 2018)	7.4 × 10^6	7.4 × 10^6	14.8 × 10^6	28.2 MB	7.4 × 10^6	7.4 × 10^6	14.8 × 10^6	28.2 MB
BT-TPF(our model)	9.2 × 10^2	4.2 × 10^3	8.4 × 10^3	3.6 KB	7.9 × 10^2	4.1 × 10^3	8.2 × 10^3	3.1 KB

Table 12
Comparison results with LNN model.

Model	UNSWNB15				Bot-IoT			
	Params	Flops	MAdds	Model Size	Params	Flops	MAdds	Model Size
LNN (Zhao et al., 2021)	5 × 10^3	5.9 × 10^4	–	190 KB	4.2 × 10^3	2.4 × 10^4	–	181 KB
BT-TPF(our model)	8.0 × 10^2	4.1 × 10^3	8.3 × 10^3	3.2 KB	7.9 × 10^2	4.1 × 10^3	8.1 × 10^3	2.9 KB

On one hand, the dimensions of image data are usually higher than those of network intrusion data, resulting in larger model sizes and higher computational costs for models used in computer vision compared to those used in intrusion detection. On the other hand, currently, well-known lightweight models are primarily used in the field of computer vision. Therefore, it would be unfair to only compare our model with existing lightweight models in computer vision. To address these issues, we also compared our proposed BT-TPF model with the LNN model proposed by Zhao et al. (2021), which is specifically designed for deployment on IoT devices. The lightweight performance of the LNN model was evaluated on the UNSW-NB15 and Bot-IoT datasets. Hence, we also conducted lightweight performance evaluation experiments on the UNSW-NB15 and Bot-IoT datasets for our proposed BT-TPF model. The comparative results are presented in Table 12.

From Table 12, it can be observed that compared to the LNN model proposed by Zhao et al., our proposed model has significantly lower parameter count, Model Size, and computational cost. The parameter count of our model is 8.0 × 10^2, which is only 16% of the parameter count of LNN. The Model Size is also only 2% of LNN's size, indicating that our model has much lower spatial cost for execution. Furthermore, our model has significantly lower FLOPs compared to LNN, indicating that the computational cost of our model is much lower than that of LNN.

In summary, through comparisons with well-known lightweight models and a new model specifically designed for IoT intrusion detection, it can be concluded that our proposed model is sufficiently lightweight.

7. Conclusion

This paper presents a novel intrusion detection model, the BT-TPF model, designed specifically for Internet of Things (IoT) systems. The proposed model employs an improved BERT-of-Theseus technique based on knowledge distillation to minimize the gap between the outputs of the Successor and Predecessor models. By doing so, the Successor model can acquire inter-category information from the Predecessor model, process and analyze large-scale data, and minimize the number of model parameters. To further reduce the dimensionality of the input features, the Siamese network is employed in our approach. The gradient propagation process of BERT-of-Theseus is analyzed and optimized to improve training efficiency and performance of the Successor model. Our experiments demonstrate the efficacy of our method in distilling a ViT-based model into a simpler Foolformer model, which achieves comparable performance to baseline models while employing fewer parameters and requiring less inference time. Moreover, the proposed BT-TPF model utilizes a Poolformer-based approach, where a spatial pooling operator is used as the token mixer to aggregate information from nearby tokens on average, without any learnable parameters. This further reduces the computational resources and model parameters, making the proposed intrusion detection model suitable for deployment on IoT nodes with limited computational power and enabling real-time detection. The proposed BT-TPF model is evaluated on the CIC-IDS2017 and TON_IoT datasets, and the experimental results demonstrate that it outperforms traditional deep learning models in terms of parameter scale and other evaluation metrics. Additionally, our proposed model achieves excellent anomaly detection results when compared to recent state-of-the-art intrusion detection models. Overall, the BT-TPF model provides a promising solution for lightweight intrusion detection in IoT systems, with potential for broader applications in other resource-constrained domains.

CRediT authorship contribution statement

Zhendong Wang: Conceptualization, Data curation, Formal analysis, Resources, Supervision, Project administration, Writing – review & editing. **Jingfei Li:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing. **Shuxin Yang:** Conceptualization, Writing – review & editing. **Xiao Luo:** Conceptualization, Writing – review & editing. **Dahai Li:** Conceptualization, Writing – review & editing. **Soroosh Mahmoodi:** Conceptualization, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China [grant numbers 62062037, 61562037, 72261018], the Natural

Science Foundation of Jiangxi Province [grant numbers 20212BAB202014, 20171BAB202026].

References

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys & Tutorials, 22*(3), 1646–1685.

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access, 8*, 165130–165150.

Asadi, M. (2022). Detecting IoT botnets based on the combination of cooperative game theory with deep and machine learning approaches. *Journal of Ambient Intelligence and Humanized Computing, 13*(12), 5547–5561.

Ashraf, J., Keshk, M., Moustafa, N., Abdel-Basset, M., Khurshid, H., Bakhshi, A. D., & Mostafa, R. R. (2021). IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. *Sustainable Cities and Society, 72*, Article 103041.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv: 1607.06450.

Bakalos, N., Voulodimos, A., Doulamis, N., Doulamis, A., Ostfeld, A., Salomons, E., … Li, P. (2019). Protecting water infrastructure from cyber and physical threats: Using multimodal data fusion and adaptive deep learning to monitor critical systems. *IEEE Signal Processing Magazine, 36*(2), 36–48.

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., & Den Hartog, F. T. (2021). ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet of Things Journal, 9*(1), 485–496.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., … Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

De Souza, C. A., Westphall, C. B., & Machado, R. B. (2022). Two-step ensemble approach for intrusion detection and identification in IoT and fog computing environments. *Computers & Electrical Engineering, 98*, Article 107694.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805.

Ding, Q., & Li, J. (2022). AnoGLA: An efficient scheme to improve network anomaly detection. *Journal of Information Security and Applications, 66*, Article 103149.

Ding, W., Abdel-Basset, M., & Mohamed, R. (2023). DeepAK-IoT: An effective deep learning model for cyberattack detection in IoT networks. *Information Sciences, 634*, 157–171.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Escorcia-Gutierrez, J., Gamarra, M., Leal, E., Madera, N., Soto, C., Mansour, R. F., … Gupta, D. (2023). Sea turtle foraging algorithm with hybrid deep learning-based intrusion detection for the internet of drones environment. *Computers and Electrical Engineering, 108*, Article 108704.

Gad, A. R., Nashat, A. A., & Barkat, T. M. (2021). Intrusion detection system using machine learning for vehicular ad hoc networks based on ToN-IoT dataset. *IEEE Access, 9*, 142206–142217.

Gautam, S., Henry, A., Zuhair, M., Rashid, M., Javed, A. R., & Maddikunta, P. K. R. (2022). A composite approach of intrusion detection systems: Hybrid RNN and correlation-based feature optimization. *Electronics, 11*(21), 3529.

Gong, Y., Liu, L., Yang, M., & Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Ho, C. M. K., Yow, K. C., Zhu, Z., & Aravamuthan, S. (2022). Network intrusion detection via flow-to-image conversion and vision transformer classification. *IEEE Access, 10*, 97780–97793.

Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).

Jayalaxmi, P. L. S., Kumar, G., Saha, R., Conti, M., Kim, T. H., & Thomas, R. (2022). DeBot: A deep learning-based model for bot detection in industrial internet-of-things. *Computers and Electrical Engineering, 102*, Article 108214.

Kasinathan, P., Pastrone, C., Spirito, M. A., & Vinkovits, M. (2013). Denial-of-Service detection in 6LoWPAN based Internet of Things. In *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)* (pp. 600–607). IEEE.

Kilincer, I. F., Ertam, F., Sengur, A., Tan, R. S., & Acharya, U. R. (2023). Automated detection of cybersecurity attacks in healthcare systems with recursive feature elimination and multilayer perceptron optimization. *Biocybernetics and Biomedical Engineering, 43*(1), 30–41.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kumar, P., Gupta, G. P., & Tripathi, R. (2021). A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *Journal of Ambient Intelligence and Humanized Computing, 12*, 9555–9572.

Li, J., Zhang, H., Liu, Y., & Liu, Z. (2022). Semi-supervised machine learning framework for network intrusion detection. *The Journal of Supercomputing, 78*(11), 13122–13144.

Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications, 36*(1), 16–24.

Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV) (pp. 116-131).

Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2021). Federated-learning-based anomaly detection for IoT security attacks. *IEEE Internet of Things Journal, 9*(4), 2545–2554.

Moustafa, N. (2019, October). New generations of internet of things datasets for cybersecurity applications based machine learning: TON_IoT datasets. In Proceedings of the eResearch Australasia Conference, Brisbane, Australia (pp. 21-25).

Moustafa, N. (2021a). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society, 72*, Article 102994.

Moustafa, N. (2021b). A systemic IoT–fog–cloud architecture for big-data analytics and cyber security systems: A review of fog computing. *Secure Edge Computing*, 41–50.

Moustafa, N., Ahmed, M., & Ahmed, S. (2020). Data analytics-enabled intrusion detection: Evaluations of ToN_IoT linux datasets. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 727–735). IEEE.

Moustafa, N., Keshky, M., Debiez, E., & Janicke, H. (2020). Federated TON_IoT Windows datasets for evaluating AI-based security applications. In *In 2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)* (pp. 848–855). IEEE.

Panagiotis, F., Taxiarxchis, K., Georgios, K., Maglaras, L., & Ferrag, M. A. (2021). Intrusion detection in critical infrastructures: A literature review. *Smart Cities, 4*(3), 1146–1157.

Roy, S., Li, J., Choi, B. J., & Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems, 127*, 276–285.

Safaldin, M., Otair, M., & Abualigah, L. (2021). Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing, 12*, 1559–1576.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (Pp. 4510-4520).

Sarhan, M., Layeghy, S., Moustafa, N., Gallagher, M., & Portmann, M. (2022). Feature extraction for machine learning-based intrusion detection in IoT networks. *Digital Communications and Networks*.

Shahin, M., Chen, F. F., Bouzary, H., Hosseinzadeh, A., & Rashidifar, R. (2022). A novel fully convolutional neural network approach for detection and classification of attacks on industrial IoT devices in smart manufacturing systems. *The International Journal of Advanced Manufacturing Technology, 123*(5–6), 2017–2029.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp, 1*, 108–116.

Sheikh, T. U., Rahman, H., Al-Qahtani, H. S., Hazra, T. K., & Sheikh, N. U. (2019). Countermeasure of attack vectors using signature-based IDS in IoT environments. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 1130–1136). IEEE.

Shukla, A. K. (2021). Detection of anomaly intrusion utilizing self-adaptive grasshopper optimization algorithm. *Neural Computing and Applications, 33*(13), 7541–7561.

Siddiqui, A. J., & Boukerche, A. (2021). TempoCode-IoT: Temporal codebook-based encoding of flow features for intrusion detection in Internet of Things. *Cluster Computing, 24*, 17–35.

Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., & Markakis, E. K. (2020). A survey on the internet of things (IoT) forensics: Challenges, approaches, and open issues. *IEEE Communications Surveys & Tutorials, 22*(2), 1191–1221.

Su, J., Vasconcellos, D. V., Prasad, S., Sgandurra, D., Feng, Y., & Sakurai, K. (2018). Lightweight classification of IoT malware based on image recognition. In *2018 IEEE 42Nd annual computer software and applications conference (COMPSAC)* (Vol. 2, pp. 664–669). IEEE.

Sun, S., Cheng, Y., Gan, Z., & Liu, J. (2019). Patient knowledge distillation for Bert model compression. arXiv preprint arXiv:1908.09355.

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105–6114). PMLR.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (Pp. 2820-2828).

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., & Lin, J. (2019). Distilling task-specific knowledge from Bert into simple neural networks. arXiv preprint arXiv:1903.12136.

Tung, F., & Mori, G. (2019). Similarity-preserving knowledge distillation. In Proceedings of the IEEE/CVF international conference on computer vision (Pp. 1365-1374).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*.

Wang, Z., Li, Z., He, D., & Chan, S. (2022). A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning. *Expert Systems with Applications, 206*, Article 117671.

Wang, Z., Liu, D., Sun, Y., Pang, X., Sun, P., Lin, F., … Ren, K. (2022). A survey on IoT-enabled home automation systems: Attacks and defenses. *IEEE Communications Surveys & Tutorials*.

Wu, Z., Zhang, H., Wang, P., & Sun, Z. (2022). RTIDS: A robust transformer-based approach for intrusion detection system. *IEEE Access, 10*, 64375–64387.

Xu, C., Zhou, W., Ge, T., Wei, F., & Zhou, M. (2020). Bert-of-Theseus: Compressing Bert by progressive module replacing. arXiv preprint arXiv:2002.02925.

Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., ... & Yan, S. (2022). Metaformer is actually what you need for vision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10819-10829).

Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications, 84*, 25–37.

Zhang, H., Huang, L., Wu, C. Q., & Li, Z. (2020). An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks, 177*, Article 107315.

Zhao, R., Gui, G., Xue, Z., Yin, J., Ohtsuki, T., Adebisi, B., & Gacanin, H. (2021). A novel intrusion detection method based on lightweight neural network for internet of things. *IEEE Internet of Things Journal, 9*(12), 9960–9972.

Zhu, J., Tang, S., Chen, D., Yu, S., Liu, Y., Rong, M., ... & Wang, X. (2021). Complementary relation contrastive distillation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9260-9269).