Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3539919, 16 pages
https://doi.org/10.1155/2022/3539919

*Research Article*

# Lightweight Traffic Classification Model Based on Deep Learning

**Chongxin Sun**,[1,2] **Bo Chen**,[1,2] **Youjun Bu**,[1,2] **Surong Zhang**,[1,2] **Desheng Zhang**,[1,2] **and Bingbing Jiang**[2]

[1]*Information Technology Institute, PLA Strategic Support Force Information Engineering University, Zhengzhou 450000, China*
[2]*Endogenous Safety and Security Research Center, Purple Mountain Laboratory, Nanjing 211100, China*

Correspondence should be addressed to Youjun Bu; buyoujun@pmlabs.com.cn

Received 27 May 2022; Revised 16 August 2022; Accepted 10 September 2022; Published 10 October 2022

Academic Editor: Jehad Ali

The development of mobile computing and the Internet of Things (IoT) has led to a surge in traffic volume, which creates a heavy burden for efficient network management. The network management requires high computational overheads to make traffic classification, which is even worse when in edge networks; existing approaches sacrifice the efficiency to obtain high-precision classification results, which are no longer suitable for limited resources edge network scenario. Given the problem, existing traffic classification generally has huge parameters and especially computational complexity. We propose a lightweight traffic classification model based on the Mobilenetv3 and improve it for an ingenious balance between performance and lightweight. Firstly, we adjust the model scale, width, and resolution to substantially reduce the number of model parameters and computations. Secondly, we embed precise spatial information on the attention mechanism to enhance the traffic flow-level feature extraction capability. Thirdly, we use the lightweight multiscale feature fusion to obtain the multiscale flow-level features of traffic. Experiments show that our model has excellent classification accuracy and operational efficiency. The accuracy of the traffic classification model designed in our work has reached more than 99.82%, and the parameter and computation amount are significantly reduced to 0.26 M and 5.26 M. In addition, the simulation experiments on Raspberry Pi prove the proposed model can realize real-time classification capability in the edge network.

## 1. Introduction

As an important task in the field of network management and network security, traffic classification is an important technical support for network control, network planning, intrusion detection, and traffic trend analysis [1–6]. With the rapid development and application surge of mobile computing and the Internet of Things, the data generated at the edge network is increasing significantly. The demand for processing and analysis of traffic at the edge network nodes is rising sharply, which brings severe challenges [7, 8]. However, most existing models lack the consideration of model efficiency, where they neither take into account the time and space complexity nor fully evaluate the efficiency. In particular, for deep learning methods, the huge overhead of memory and runtime in the complex neural network leads to high energy consumption, which is not feasible for edge devices with limited resources [9–11].

In this paper, we propose a lightweight traffic classification model based on deep learning to raise the efficiency (especially computational efficiency) and performance of traffic classification at the limited resources edge network. And the main contributions of this paper can be summarized as follows:

(1) We use a novel and outstanding lightweight model Mobilenetv3 as the basic model of our model to reduce the time and space complexity. In addition, we compress the scale, width, and resolution of Mobilenetv3 to minimize parameters and computation effectively of our model, which makes it high-speed traffic classify in edge networks

(2) We fuse the spatial attention mechanism containing precise location information on the original channel attention mechanism of Mobilenetv3 to enhance the spatial feature (high-order flow-level traffic feature)

extraction ability of the traffic classification model at a small cost

(3) We embed multiscale feature extraction structures to replace the original feature extraction module of Mobilenetv3 for more high-level semantic information and detailed information of traffic. At the same time, we referred the idea of Mobilenetv3 to further lightweight the multiscale feature fusion module

The rest of this paper is structured as follows: Section 2 introduces the related work of traffic classification. The systematic work is presented in Section 3, Section 4 elaborates the experimental setup, and Section 5 launches the analysis of experimental results. Finally, Section 6 concludes this paper.

## 2. Related Work

Despite several years of development, there are still considerable deficiencies in traffic classification methods based on machine learning. Most existing methods based on machine learning deeply rely on traffic statistical features such as average packet length, flow duration, and average arrival time of packets [10, 11]. The extraction of statistical features needs to observe the whole or most of the flow, so machine learning is mainly used in offline classification scenarios [12]. In addition, feature selection, as the core of the machine learning method, heavily relies on expert experience. For example, Moore et al. designed 250 features based on prior knowledge [13]. Then most subsequent traffic classification works of machine learning mainly were based on these 250 features [14, 15], which led to the performance of traffic classification depending on the rationality of feature selection.

Compared with machine learning, deep learning can automatically extract different types of network traffic features by using neural networks, which solves the problem of heavily relying on expert experience in machine learning methods [16]. The traffic classification model based on deep learning can directly learn and output the corresponding classification results by inputting the original traffic data. Wang et al. [16] first proposed an end-to-end traffic classification method based on a convolutional neural network (CNN), which integrates feature extraction, feature selection, and classifier into a unified framework to learn the nonlinear relationship between original input and expected output automatically, bringing a new direction to traffic classification. Then, traffic classification methods based on deep learning emerge in endlessly. Shapira and Shavitt [17] proposed a new traffic classification and application recognition method, which converts the traffic data into intuitive images and then uses CNN for classification. Wang et al. [18] used CNN with deviation standardization to classify network traffic on the Moore dataset, which achieved good classification performance. Lopez-Martin et al. [19] combined CNN and recurrent neural network (RNN) to classify traffic. Zou et al. [20] used bidirectional long short-term memory (BLSTM) to extract the forward and backward features of byte sequences in a session to extract traffic features more comprehensively. Although the above work has achieved good results, it brings new problems: most existing deep learning models lack thinking about efficiency [16, 21]. The performance improvement is often accompanied by the increasing depth and complexity of the model, which brings practical application problems: complex neural networks with up to gigabytes of memory usage and high computation costs [22, 23] making it difficult to be applied to edge terminal devices with limited hardware resources, which seriously limits the application of traffic classification technology based on deep learning.

Therefore, the lightweight deep learning model has gradually become the research focus. Roy et al. [24] improved on CNN-LSTM and proposed a traffic classification method based on OdeNet-LSTM, which achieved faster reasoning speed. Fauvel et al. [25] introduced a new residual structure design into the CNN-based traffic classification method and proposed LexNet, significantly reducing the parameter. Liu et al. [26] lightened the model by replacing the LSTM in work [20] with a gated recurrent unit (GRU) [27] and used the attention mechanism to allocate weights according to the contribution of features to classification, which is called BGRUA. Similar lightweight traffic detection methods are also available in works [28, 29]. Although the above studies have achieved good results, they all focus on the reducing model parameter, ignore the weak computing capabilities of most edge devices, resulting in difficulties in the actual operation of the above models in edge networks and weak real-time traffic classification.

## 3. System Model Design

We select the Mobilenetv3-Small [30] as the backbone and improve it through scale, width factor, resolution ratio adjustment, coordinate attention mechanism (CA), and multiscale feature fusion to build a low-cost and high-precision traffic classification model. The improved network structure is shown in Figure 1, where Bottleneck-CA is the bottleneck with the coordinate attention mechanism, Bottleneck-Multi is the bottleneck with multiscale feature fusion structure, $S$ is the step size of convolution, and DW and PW are depthwise convolution and pointwise convolution, respectively. The coordinate attention mechanism is embedded when the stride is 1, and the bottleneck is replaced with a multiscale feature fusion structure when the stride is 2.

*3.1. Mobilenetv3.* Compared with other lightweight models, the biggest advantage of Mobilenetv3 is that computational complexity is extremely small and can be flexibly deployed in various devices, which is suitable for edge devices. Hence, we use a lightweight convolutional neural network Mobilenetv3-Small as the basic model to solve the problem that mobile devices are challenging to run complex deep learning models due to the limited hardware resources.

The excellent performance of Mobilenetv3 benefits from the following: (1) use depthwise separable convolution

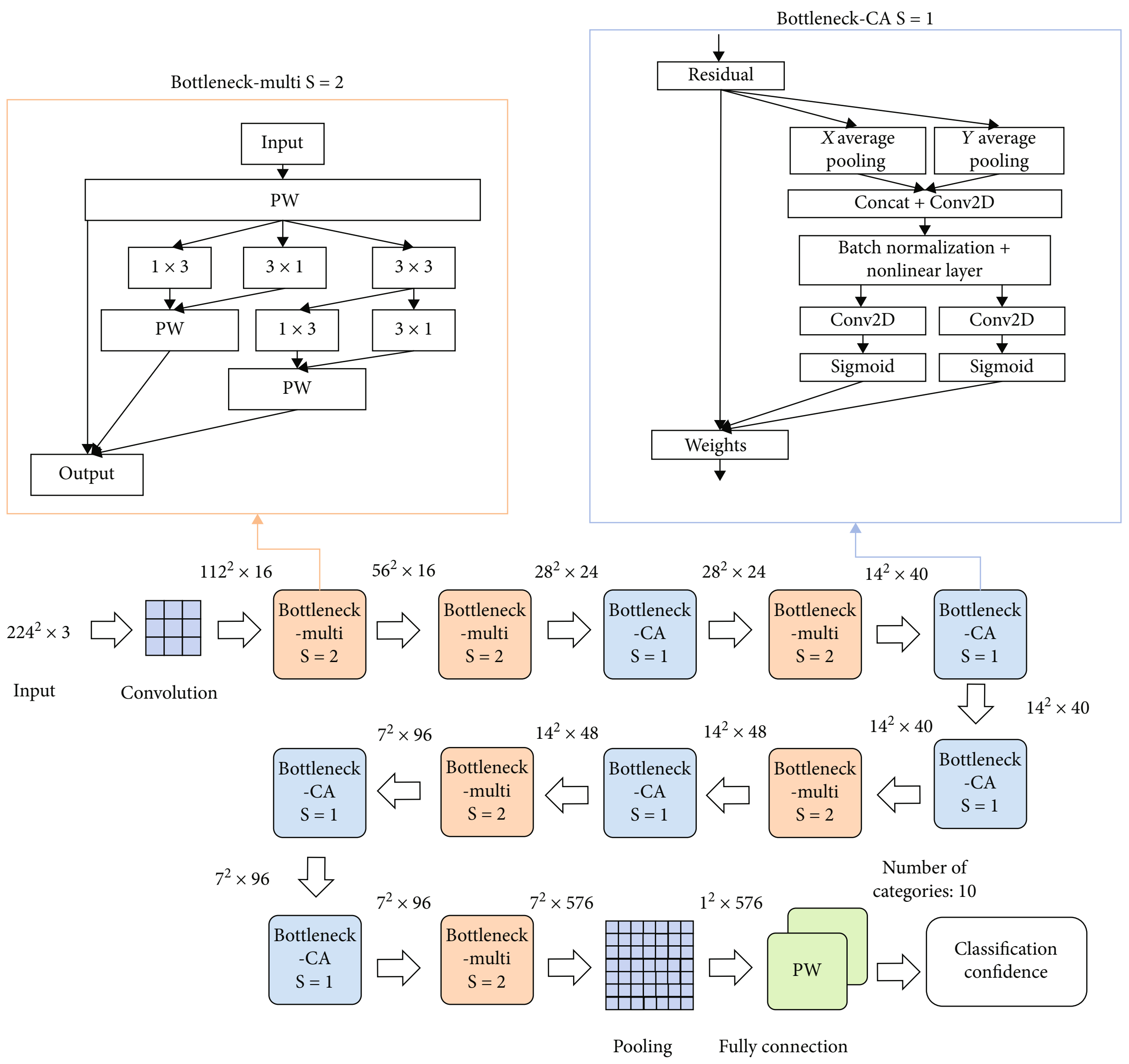


Figure 1: Overall network framework of the proposed model.

instead of conventional convolution operation to reduce the number of parameters and computations, (2) use linear bottleneck structure to reduce the loss of low dimensional feature information and the reverse residual structure to deepen the network and enhance the ability of feature expression for better classification performance, (3) provide different model scales (large and small), widths, and resolutions for flexible parameter and computation adjustment.

Depthwise separable convolution shown in Figure 2 divides conventional convolution operations into depthwise convolution (DW) and pointwise convolution (PW). This operation of decomposing conventional convolution into two steps not only makes its performance equivalent to the conventional convolution but also can effectively reduce the number of computations and model parameters. The specific principles are as follows:

Assume that the dimension of the input feature is $D_f \times D_f \times M$, the size of the convolution kernel is $D_k \times D_k$, the dimension of the output feature is $D_f \times D_f \times N$, and $M$ and $N$ are the numbers of input and output channels.

The computation amount of using standard convolution is

$$\mathrm{MAC1} = D_k \times D_k \times M \times N \times D_f \times D_f. \tag{1}$$

The computation amount of using depthwise separable convolution is

$$\begin{aligned}\mathrm{MAC2} = {} & \alpha D_k \times D_k \times M \times \beta D_f \times \beta D_f \\ & + \alpha^2 M \times N \times \beta D_f \times \beta D_f.\end{aligned} \tag{2}$$

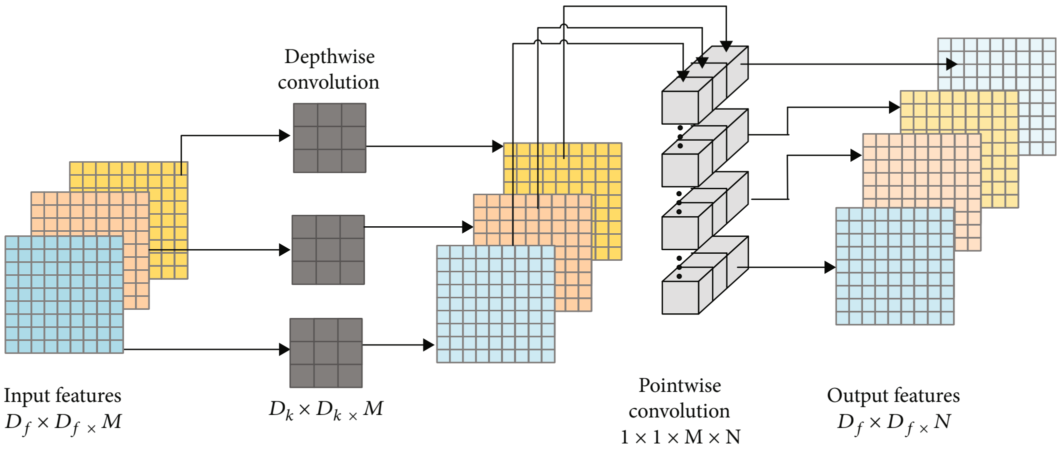

Depthwise convolution
Input features $D_f \times D_f \times M$
$D_k \times D_k \times M$
Pointwise convolution $1 \times 1 \times M \times N$
Output features $D_f \times D_f \times N$

Figure 2: Principle of deep separable convolution.

The computation ratio of depthwise separable convolution and standard convolution is

$$\frac{\text{MAC2}}{\text{MAC}_1} = \frac{\alpha D_k \times D_k \times M \times \beta D_f \times \beta D_f + \alpha^2 M \times N \times \beta D_f \times \beta D_f}{D_k \times D_k \times M \times N \times D_f \times D_f} = \frac{\alpha\beta^2}{N} + \frac{\alpha^2\beta^2}{D_k^2}, \quad (3)$$

where $\alpha$ is the width factor, and common configurations are 0.25, 0.5, 0.75, and 1. $\beta$ is the resolution factor; commonly used configurations are 4/7, 5/7, 6/7, and 1. It can be seen from formula (3) that a single depthwise separable convolution can save the amount of computation compared with the standard convolution $((\alpha\beta^2)/(N)) + ((\alpha^2\beta^2)/(D_k^2))$.

The linear bottleneck structure shown in Figure 3 firstly uses PW to increase the dimension, then uses DW to extract features, then uses the channel attention mechanism to accurately model the relationship between each channel of the convolution feature, and then uses PW to reduce the dimension. Finally, it uses a linear activation function and the reverse residual structure to reduce the feature loss. The inverse residual structure hardly increases the parameter and computation amount and comprehensively improves the searchability, thereby effectively improving the performance. It is worth noting that the linear bottleneck structure uses the reverse residual structure only when the convolution step is one.

*3.2. Data Preprocessing.* We draw on the data preprocessing method in work [17], which adopts the following five steps to convert the original traffic data stored in Pacp format into an image format to facilitate the processing of the model, as shown in Figure 4.

*(1) Flow Segmentation.* Divide the original traffic Pacp files into different bidirectional sessions according to the source IP, destination IP, source port, destination port, and transport layer protocol

*(2) Data Cleaning.* Delete duplicate and empty traffic packets, iterate over all data packets of bidirectional sessions and delete information unrelated to traffic classification, such as MAC addresses

*(3) Unified Length.* Unify the session length is 784 bytes. If the session length is more than 784 bytes, it will be truncated; if less than 784 bytes, zeros bytes will be added at the end of the session. In addition, padding zeros bytes at the end of the header (8 bytes) of the UDP segment to equal the length of the TCP header (20 bytes) makes the transport layer segment uniform

*(4) Data Visualization.* Convert each session (784 bytes) of uniform length into a two-dimensional grayscale image with a resolution of 28∗28. Bytes 0x00 to 0xff in the session correspond to pixel values in the grayscale chromaticity interval from black to white

*(5) Data Enhancement.* The fact that some traffic categories have a few samples may cause a phenomenon of data skew. To avoid the overfitting problem caused by the over-reliance on a small number of samples, we perform image enhancement on traffic categories with a small number of samples, including random flipping, mirroring, Gaussian noise, and filtering to increase the number of samples

*3.3. The Coordinate Attention Mechanism.* The Squeeze-and-excitation module (SE) of Mobilenetv3 introduces the channel attention mechanism, improving the classification accuracy by assigning weights to channels of different importance. However, the channel features of the grayscale traffic images are not prominent, and there is no strong correlation between the adjacent channels. Therefore, the channel attention mechanism is not suitable for traffic image classification.

The traffic characteristics determine that the fields in different locations of the traffic images represent different

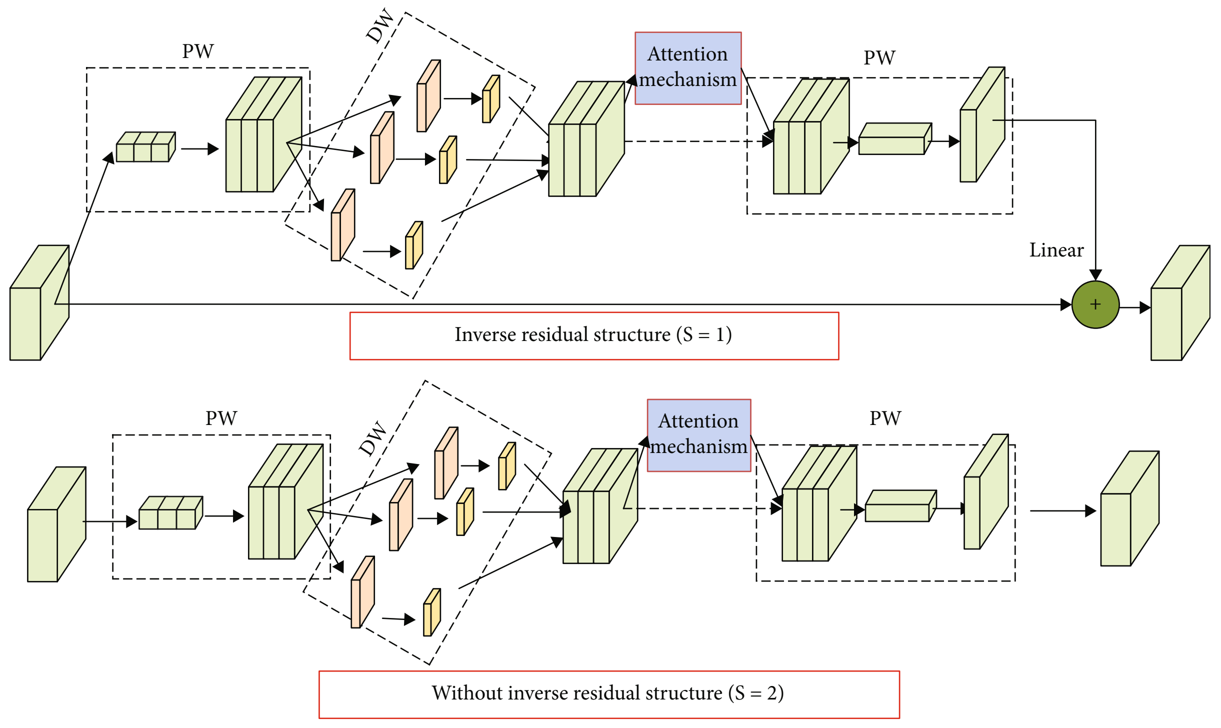

PW
DW
Attention mechanism
PW
Linear
+
Inverse residual structure (S = 1)
PW
DW
Attention mechanism
PW
Without inverse residual structure (S = 2)

FIGURE 3: Bottleneck structure under two strides.

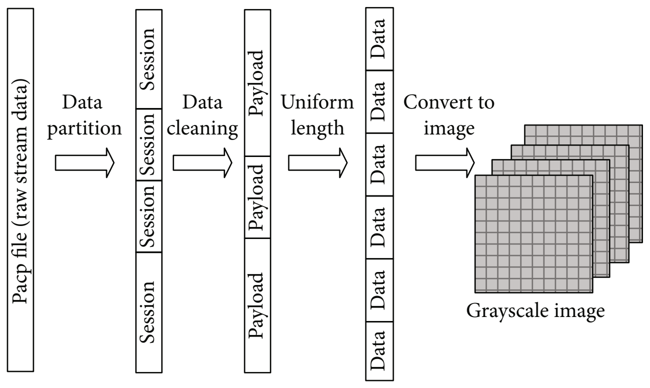


FIGURE 4: Data preprocessing.

information (flow-level feature), which have different degrees of importance in traffic classification. For example, the first 20 bytes in the TCP traffic header represent application protocol information, which is helpful for traffic identification. According to work [17], the payload information between traffic classes strongly correlates with the field position, and the payload information of traffic categories has different features in different situations. It can be said that location information is crucial for the classification of traffic images. And the SE module compresses the global spatial information into the channel descriptor, leading to the loss of position information.

Therefore, we referred the coordinate attention mechanism (CA) [31] to embed the spatial scale based on the SE module, which decomposes the global average pooling of the SE module into one-dimensional pooling of the horizontal and vertical directions to obtain the relevant information on horizontal direction X and vertical direction Y, making our model can precisely capture the location information to improve the traffic classification ability. The comparison between SE and CA is shown in Figure 5. The specific location of the replacement (the reverse residual structure only when the convolution step is one) is shown in Figure 1.

Firstly, we decompose the global average pooling of SE module to generate a one-dimensional perceptual attention feature on X and Y.

$$Z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} xc(i, j), \tag{4}$$

$$Z_c^h = \frac{1}{W} \sum_{0 \le i \le W} x_c(h, i), \tag{5}$$

$$Z_c^w = \frac{1}{H} \sum_{0 \le j \le H} x_c(j, w), \tag{6}$$

where $C$ is the number of channels, $H$ and $W$ are the height and width of the input feature, respectively, $Z$ is the output, $x_c$ is the two-dimensional feature set of the $c$-th channel, $i$ and $j$ represent the coordinates on the output feature, and $w$ and $h$ are the convolution kernel weight of the $c$-th channel.

Secondly, we concatenate the one-dimensional features of X and Y and send them to the transformation function F1 to generate the intermediate feature $f$ containing horizontal and vertical spatial information.

$$f = \partial\left(F1\left(\left[z^h, z^w\right]\right)\right), \tag{7}$$

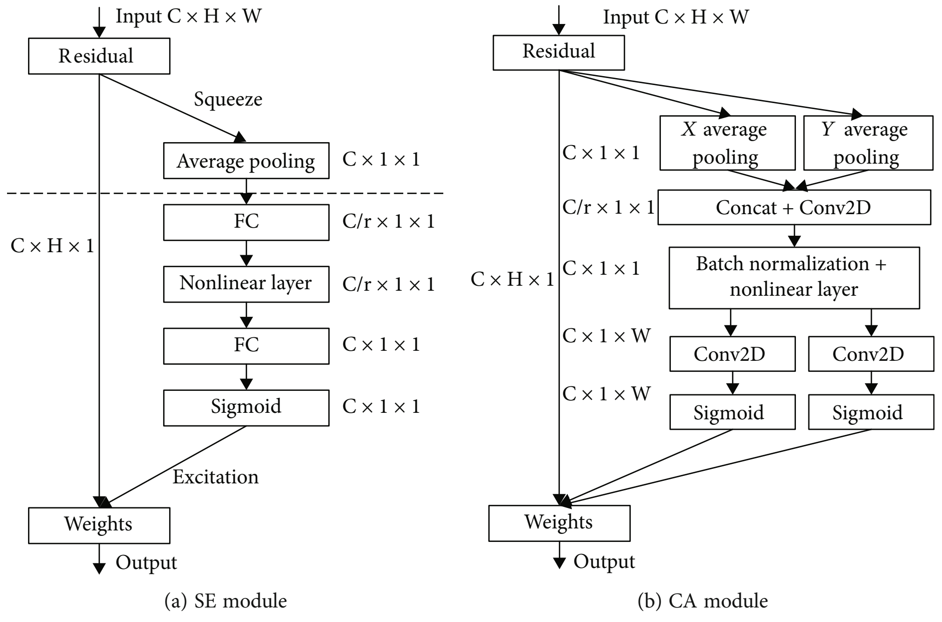


FIGURE 5: Coordinate attention mechanism.

where [,] is the concatenation operation, $f \in R^{c/r\times(H+W)}$,$\partial$ is the nonlinear activation function, and $r$ is a hyperparameter that controls the size of the module.

Thirdly, we decompose $f$ into two separate tensors $f^h \in R^{c/r\times H}$ and $f^w \in R^{c/r\times W}$, and then use convolution transformation functions $F_h$ and $F_w$ to transform $f_h$ and $f_w$ into tensors with the same number of channels of input feature $g^h$ and $g^w$.

$$g^h = \sigma\left(F_h\left(f^h\right)\right)\mathrm{g}^{\mathrm{h}} = \delta\left(\mathrm{F}_{\mathrm{h}}\left(\mathrm{f}^{\mathrm{h}}\right)\right), \tag{8}$$

$$g^w = \sigma(F_w(f^w))\mathrm{g}^{\mathrm{w}} = \delta(\mathrm{F}_{\mathrm{w}}(\mathrm{f}^{\mathrm{w}})). \tag{9}$$

Among them, $\delta$ is the sigmoid function.

Finally, we multiply the input feature by the horizontal weight (formula (8)) and the vertical weight (formula (9)) to obtain the feature of the coordinate attention output

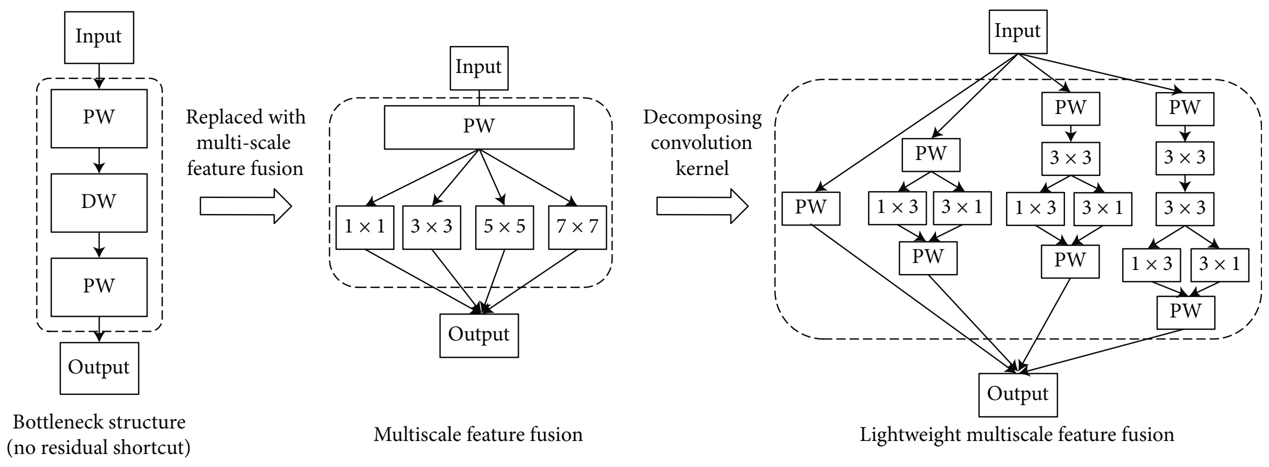
$$y_c(i,j) = x_c(i,j) \times g_c^h(i) \times g_c^w(j). \tag{10}$$

*3.4. The Multiscale Feature Fusion.* From the traffic visualization results in Section 4.2, there are different scale characteristics between traffic categories, such as the traffic texture scale of DOS-UDP is more extensive than that of the Service-scan. If the small visual receptive field is used for feature extraction, some higher-level semantic information will be ignored. On the contrary, large-scale feature extraction will get higher-level semantic information, which will lead to the loss of spatial geometric feature details of traffic images. The multiscale feature extraction module uses different scales to process the traffic image, which enables the model to understand the traffic image more comprehensively and makes the spatial scale extracted by the feature more abundant, which is conducive to the subsequent traffic classification. As shown in Figure 6, the multiscale feature fusion module uses $1 \times 1$, $3 \times 3$, $5 \times 5$, and $7 \times 7$ convolution kernel to parallel multibranch extract traffic image features to enhance the spatial scale adaptability of the model in traffic feature extraction, for enhancing the recognition ability of the proposed model to traffic texture features and improve the classification performance.

Therefore, based on work [32], we propose the multiscale feature fusion module that integrates multiple scales and replaces the linear bottleneck structure without reverse residual structure in the original model with the multiscale feature fusion module shown in Figure 6. The replacement position (the reverse residual structure only when the step is two) is shown in Figure 1. At the same time, in order to reduce the weight of the multiscale feature extraction module as much as possible, we draw on the idea of depthwise separable convolution to decompose the $5 \times 5$ and $7 \times 7$ convolution of the multiscale feature extraction module into serial $3 \times 3$ convolutions and further decomposes the last $3 \times 3$ convolution to a parallel $1 \times 3$ and $3 \times 1$ convolution, which fully reduces the computation and memory consumption brought by this module.

## 4. Experimental Setup

*4.1. Experimental Environment.* The specific experimental environment and configuration in this paper are as follows: CPU is Intel Core I7-9700, graphics card is NVIDIA RTX 2080Ti, memory is 16GB, the operating system is Ubuntu 18.04.3, programming language is Python 3.7 and Pytorch 1.8.1 implement a traffic classification model. And the edge device used in Section 5.4 selects Raspberry Pi 4 Model B, the configuration is as follows: CPU is BCM2711 (4-core 1.5GHz), memory is 4GB, the operating system is Raspberry Pi OS 3.2, programming language and version is Python 3.7, and deep learning framework is Pytorch 1.2.



Input
PW
DW
PW
Output
Bottleneck structure (no residual shortcut)
Replaced with multi-scale feature fusion
Input
PW
1 × 1
3 × 3
5 × 5
7 × 7
Output
Multiscale feature fusion
Decomposing convolution kernel
Input
PW
PW
1 × 3
3 × 1
PW
PW
3 × 3
1 × 3
3 × 1
PW
PW
3 × 3
3 × 3
1 × 3
3 × 1
PW
Output
Lightweight multiscale feature fusion

FIGURE 6: Multiscale feature extraction module.

TABLE 1: Bot-IoT dataset.

| Traffic type | Traffic class | |
|---|---|---|
| Data collection | Service scanning, OS fingerprinting | |
| Denial of service attack | DDoS | TCP, UDP, HTTP |
| | Dos | TCP, UDP, HTTP |
| Data theft | Keylogging, data theft | |

Through theoretical and experimental analysis, we set the final model training learning rate to 0.001, the epoch to 50, each batch of training images to 128, divide the training set and the test set according to the ratio of 9 : 1, selects Adam as the model optimizer, and uses cross-entropy as the loss function.

### *4.2. Dataset and Experimental Scenario*

*4.2.1. Dataset.* We select the IoT traffic dataset Bot-IoT [33] for experiments to prove our model is suitable for edge network traffic classification, which contains ten different traffic categories involving data collection and denial of service attacks, information theft, and other scenarios, as shown in Table 1.

Figure 7 shows the visualization results of all traffic categories of the Bot-IoT, with four images randomly selected from each category. It can be seen that the traffic images between different protocols are easy to distinguish. Although the traffic images of the same protocol are generally similar, such as DOS-UDP and DDOS-UDP, the texture details are relatively different. In addition, the flow texture distribution has a certain positional regularity. In summary, it can be seen from the traffic visualization that using the CA attention mechanism and multiscale feature fusion can achieve better results in theory.

We select the encrypted traffic dataset USTC-TFC2016 dataset [34] to prove our model can classify encrypted traffic, including twenty encrypted traffic categories, as shown in Table 2.

*4.2.2. Experimental Scenario.* We set up four experimental scenarios to thoroughly verify the classification performance, as shown in Table 3. Encrypted-2 experimental scenario is the two-category problem between malicious and normal encrypted traffic; Bot-IoT-3 scenario is the three-category classification of IoT malicious traffic, including data collection, denial of service attacks, and information stealing traffic; Bot-IoT-10 scenario is the ten-category classification of IoT traffic; Encrypted-20 scenario is the encrypted traffic classification of 20-category.

### *4.3. Evaluation Metrics and Baselines*

*4.3.1. Evaluation Metrics.* To evaluate the effectiveness of our model, we adopt four generally recognized metrics, namely accuracy, precision, recall, and F1-score, accuracy represents the proportion of correctly classified flows. With respect to the other three metrics, we obtain the true positive (TP), false positive (FP), and false negative (FN) by comparing the output classification results with labeled ground truth. The information above can be used to calculate the accuracy, precision, recall, and F1-score index.

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN}, \\ \text{Precision} &= \frac{TP}{TP + FP}, \\ \text{Recall} &= \frac{TP}{TP + FN}, \\ \text{F1} - \text{score} &= \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 2. \end{aligned} \tag{11}$$

Among them, for multiclassification problems, each class, and remaining class samples are regarded as a binary classification, and the precision and recall rate of each type are directly calculated. True positive (TP), false positive (FP), false negative (FN), and true negative (TN) are obtained by comparing the output classification results with the true labels.

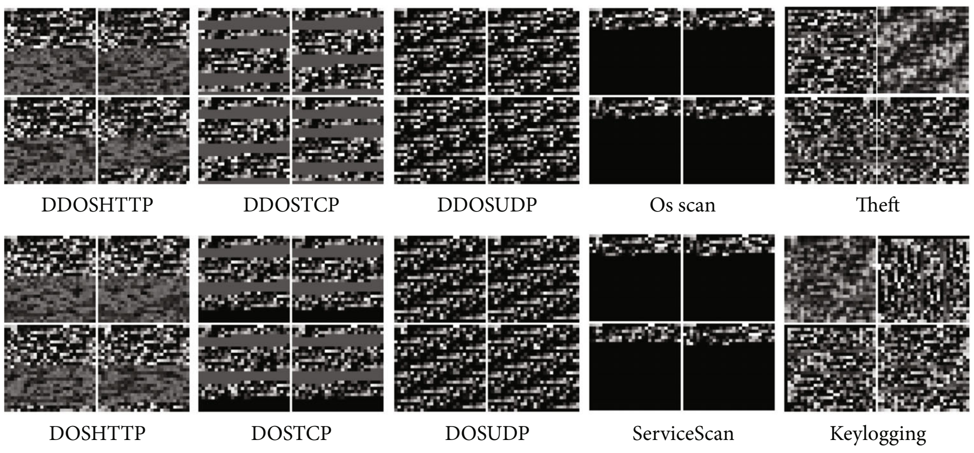


FIGURE 7: Traffic visualization.

TABLE 2: USTC-TFC2016 dataset.

| Traffic type | Traffic class |
|---|---|
| Normal encrypted traffic | BitTorrent, facetime, FTP, Gmail, MySQL, outlook, skype, SMB, Weibo, WorldOfWarcraft |
| Malicious encrypted traffic | Cridex, Geodo, Htbot, Miuref, Neris, Nsis-ay, Shifu, Tinba, Virut, Zeus |

TABLE 3: Experimental scene settings.

| Experimental scene | Content | Class | Experiment number | Specific category |
|---|---|---|---|---|
| Encrypted-2 | Encrypted traffic identification | 2 | 0 | Malicious encrypted traffic |
| | | | 1 | Normal encrypted traffic |
| Bot-IoT-3 | IoT traffic type identification | 3 | 0 | Data collection |
| | | | 1 | Denial of service attack |
| | | | 2 | Data theft |
| Bot-IoT-10 | IoT traffic class classification | 10 | 0-9 | Bot-IoT all classes |
| Encrypted-20 | Encrypted traffic class classification | 20 | 0-19 | USTC-TFC2016 all classes |

TABLE 4: Classification results for different resolution factors.

| Resolution size | Accuracy (%) | Parameter (M) | Floating-point operations (M) |
|---|---|---|---|
| 1 | 99.50 | 1.98 | 58.82 |
| 6/7 | 99.32 | 1.98 | 43.62 |
| 5/7 | 99.01 | 1.98 | 30.75 |
| 4/7 | 98.73 | 1.98 | 20.23 |

TABLE 5: Classification results for different width factors.

| Model | Accuracy (%) | Parameter (M) | Floating-point operations (M) |
|---|---|---|---|
| Small1.0 | 98.73 | 1.98 | 20.23 |
| Small0.75 | 98.53 | 1.16 | 13.97 |
| Small0.5 | 98.02 | 0.52 | 6.82 |
| Small0.25 | 97.60 | 0.15 | 3.26 |

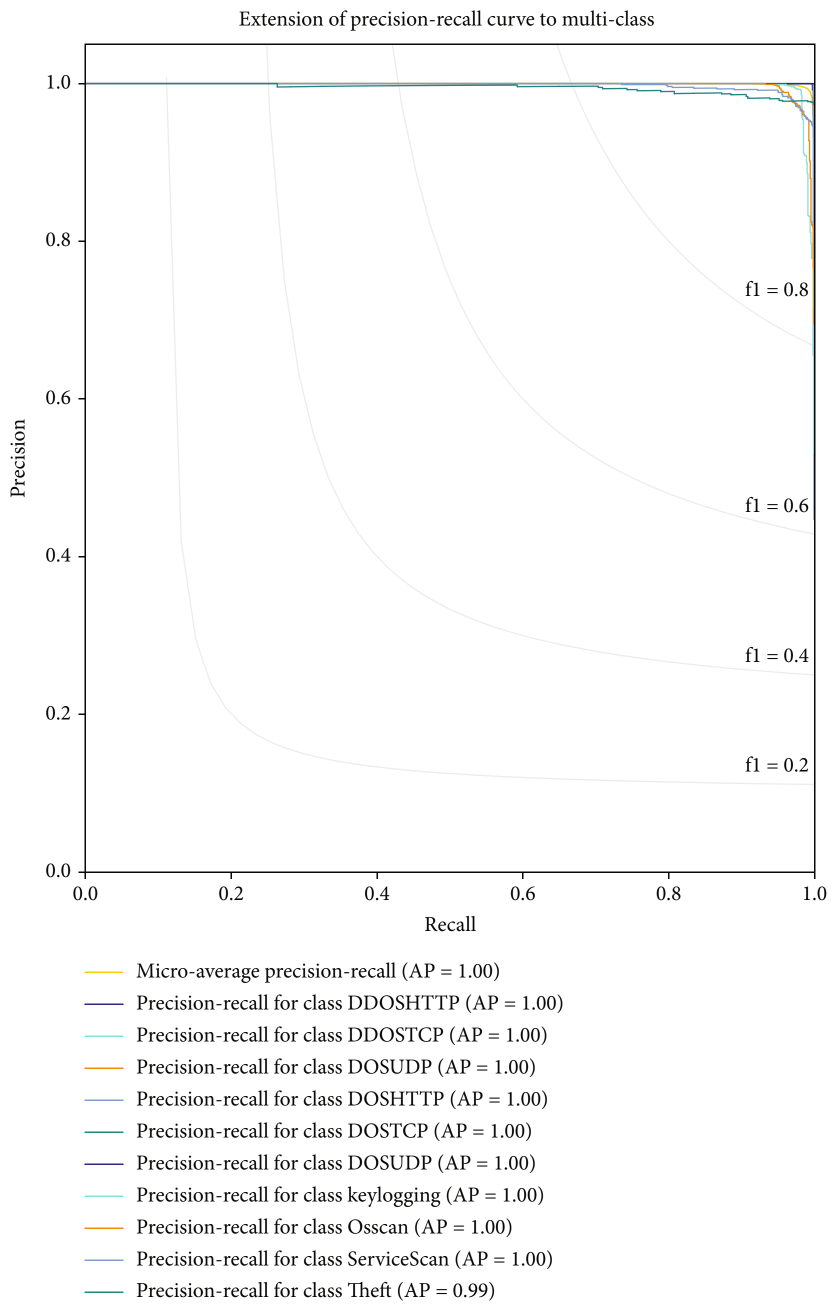
Additionally, to evaluate the efficiency of our model, we measure the parameter and computation, in which the unit of them is M. The parameter refers to how many parameters the model contains, which reflects the space complexity of the model and determines the storage space required by the model. The computation is described by FLOPs (floating-point of operations), which reflects the time complexity of the model. It refers to the number of computations required for model inference. When the hardware device is determined, it determines the model inference speed.

*4.3.2. Baselines.* To confirm the effectiveness of our model, we compare the model with the following baselines in terms of accuracy, precision, recall, and F1 score. For the efficiency, our model is compared with other models in terms of parameters and computation.

*(1) 1D-CNN* [17]. This model integrates feature selection, feature extraction, and classifier with 1D-CNN into a unified end-to-end framework, aiming to automatically learn the nonlinear relationship between the original input

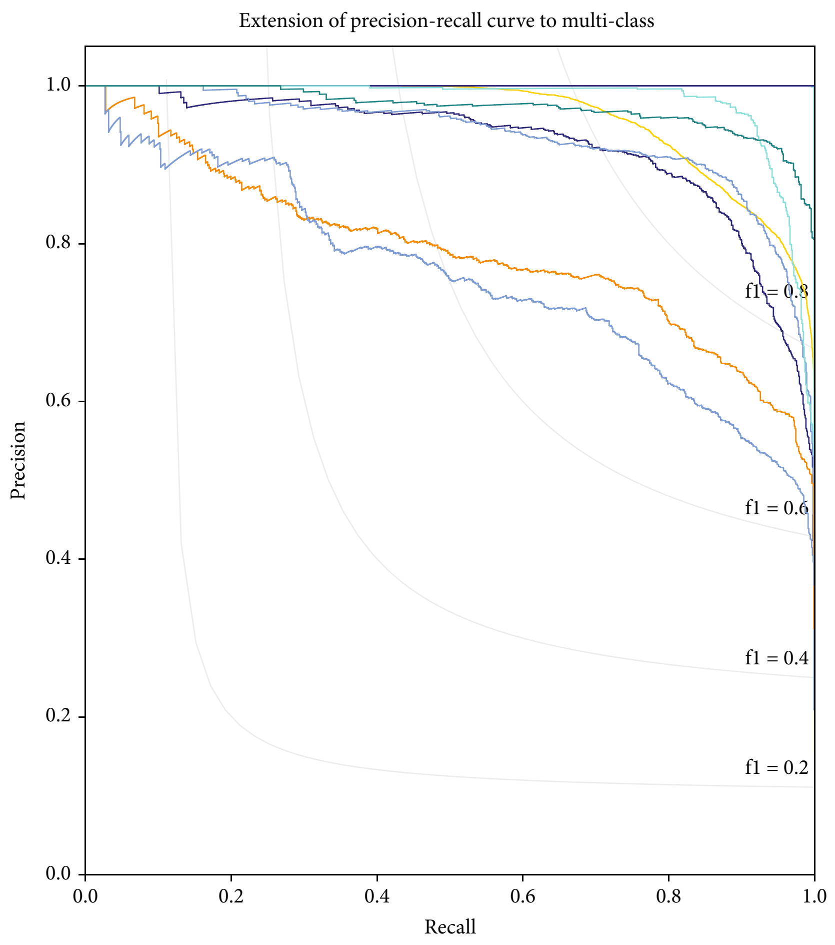
Table 6: Ablation study results in the Bot-IoT-10.

| Model | Attention mechanism | Multiscale feature extraction structure | Accuracy (%) | Parameters (M) | Floating-point operations (M) |
|---|---|---|---|---|---|
| | SE (basic model) | 0 | 97.60 | 0.15 | 3.26 |
| Contrast model | CA | 0 | 98.48 | 0.17 | 3.82 |
| | SE | 1 | 99.08 | 0.20 | 4.07 |
| Proposed model | CA | 1 | 99.82 | 0.26 | 5.26 |



(a) Proposed model

Figure 8: Continued.



Micro-average precision-recall (AP = 0.96)
Precision-recall for class DDOSHTTP (AP = 0.92)
Precision-recall for class DDOSTCP (AP = 1.00)
Precision-recall for class DOSUDP (AP = 1.00)
Precision-recall for class DOSHTTP (AP = 0.94)
Precision-recall for class DOSTCP (AP = 1.00)
Precision-recall for class DOSUDP (AP = 1.00)
Precision-recall for class keylogging (AP = 0.98)
Precision-recall for class Osscan (AP = 0.79)
Precision-recall for class ServiceScan (AP = 0.76)
Precision-recall for class Theft (AP = 0.97)

(b) Basic model

Figure 8: The precision-recall curve of our model and the basic model in the Bot-IoT-10 scenario.

and the desired output. This is the first application of an end-to-end approach in traffic classification

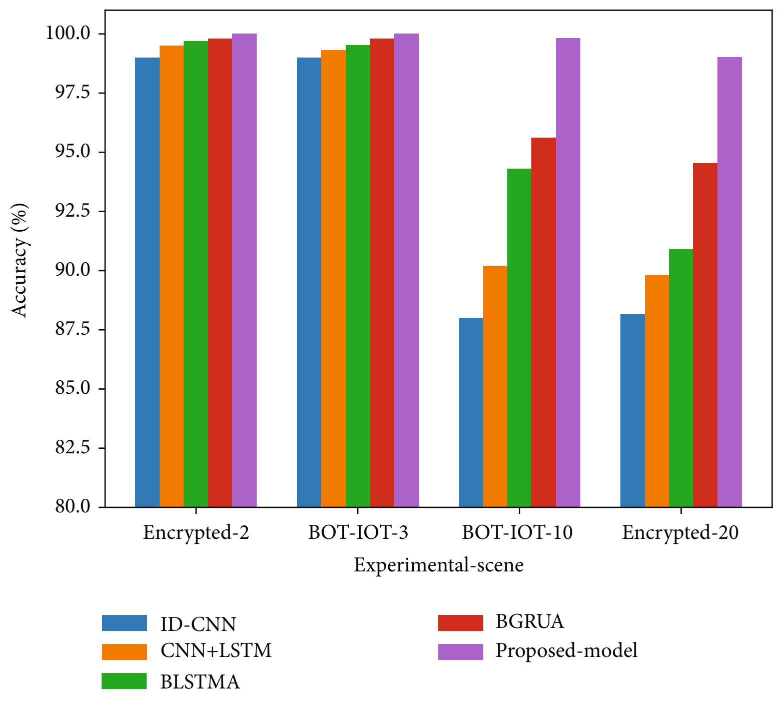
*(2) CNN-LSTM* [20]. This model combines CNN and RNN. CNN is used to extract packet features, and the LSTM network is utilized to extract traffic flow features to improve traffic classification accuracy

*(3) BLSTMA* [16]. This approach embeds an attention mechanism in a bidirectional LSTM to extract flow features and assign feature weights through an attention mechanism

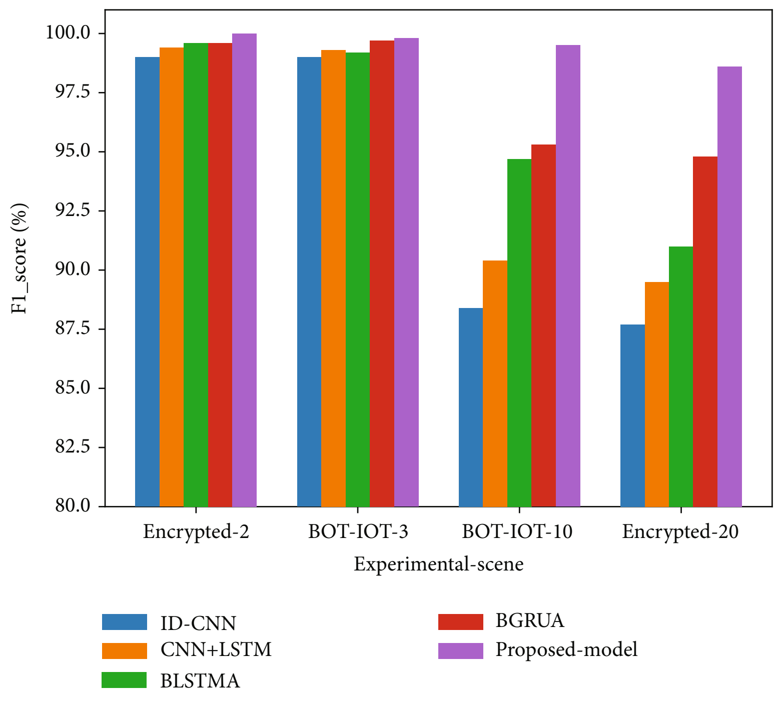
*(4) BGRUA* [28]. This model utilizes the gated recurrent unit (GRU) to model the packet interactions, and the attention mechanism is adopted to assign weights to the features according to the contribution of the traffic features to the classification. In addition, the BGRUA model is the most advanced in the lightweight traffic classification method based on deep learning at present [13].

## 5. Results and Discussion

*5.1. Selection and Analysis of Hyperparameter.* To achieve the best lightweight performance, we compress the resolution and width factor to reduce the parameters and FLOPs according to the change of the accuracy rate based on the Bot-IoT-10 experimental scene to determine the optimal

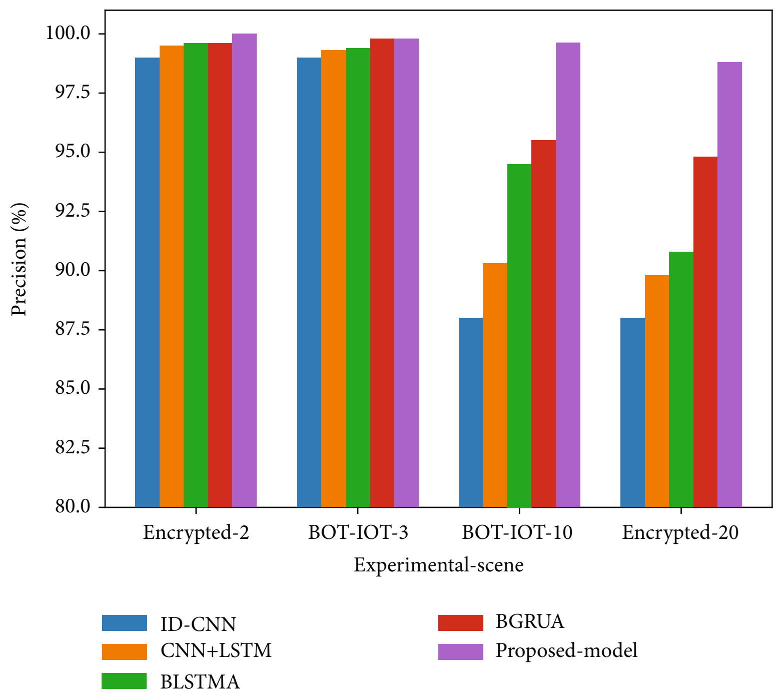


(a) Accuracy

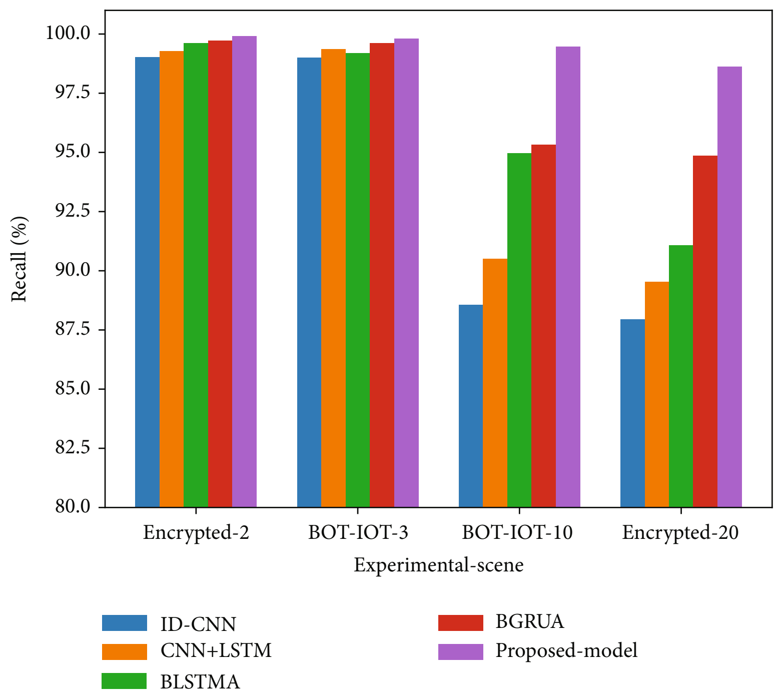


(b) F1-score

Figure 9: Continued.



(c) Precision



(d) Recall

Figure 9: Comparison of traffic classification models in four experimental scenarios.

hyperparameter so that our model can achieve the optimal balance between lightweight and accuracy.

Based on the Mobilenetv3 Small-1.0 model, we firstly change the resolutions to conduct comparative experiments to determine the resolutions factor choice. Table 4 shows the accuracy, parameters, and computation amount under the different resolution factors.

As shown in Table 4, the performance is continuously weakened with the resolution decreasing, and the amount of computation is significantly attenuated. When the resolution factor is 4/7, the model classification accuracy is 98.73%. Although the accuracy partially attenuates compared with the default resolution, the computation amount is significantly reduced by 65.6%. Especially, the number of

Table 7: Efficiency comparison of each traffic classification model in Bot-IoT-10 scenario.

| Model | Parameter (M) | Floating-point operations (M) |
|---|---|---|
| Proposed model | 0.26 | 5.26 |
| 1D-CNN | 5.83 | 1986.29 |
| CNN-LSTM | 2.39 | 7484.16 |
| BLSTMA | 0.19 | 18.66 |
| BGRUA | 0.14 | 14.08 |

parameters does not decrease with the resolution decreasing. This is because the change of the feature map will not influence the parameter but only affect the addition and multiplication computation [29]. Therefore, we select 4/7 as the resolution factor to minimize the computation amount of our model so that our model can perform real-time traffic classification on edge devices with limited computing resources.

Secondly, we fixed 4/7 resolution factors to carry out comparative experiments for the choice of width factor. Table 5 shows the accuracy, parameter quantity, and computation quantity under the four width factors.

It can be seen that as the width factor decreases, the computational complexity and parameter complexity both abate to different degrees. When the accuracy of the Mobilenetv3 Small-0.25 reaches 97.60%, the amount of computation and parameters is only 0.15 M and 3.26 M, which is far less than the default model, and only 7.57% of the amount of computation and 16.60% of the number of parameters of small 1.0 model. Therefore, we determine the width factor as 0.5 to further reduce the memory usage and computing resources of our model.

In summary, we choose Mobilenetv3 Small-0.25 with the resolution factor of 4/7 as the Basic model for traffic classification in this paper, sacrificing some classification performance to lighten the traffic classification model as much as possible.

*5.2. Ablation Experiment.* In order to verify the contribution of the coordinate attention mechanism and multiscale feature fusion structure to the improvement of model performance under the premise of slightly increasing the amount of computation and parameter, we conduct ablation experiments based on the Bot-IoT-10, with the accuracy rate, parameters, and FLOPs as evaluation metrics.

To verify the effectiveness of the coordinate attention mechanism in improving accuracy, we, respectively, embed SE and CA modules in the attention mechanism for comparison. To verify the contribution of the multiscale feature fusion structure to our model, the linear bottleneck layer that does not contain the inverse residual shortcut is replaced with the multiscale feature fusion structure for comparison. The ablation experiment results are shown in Table 6.

It can be seen that, compared with the basic model, the CA module increases the accuracy to 98.48% at a slight increase of 0.56 M computation and 0.02 M parameters, and the multiscale feature fusion structure increases the accuracy to 99.08%.

In order to more intuitively prove the merits of our improvements, we compare our model with the basic model based on the experimental scenario Bot-IoT-10. Figure 8 shows the precision-recall curve of our model and the basic model, where average precision (AP) is the area under the precision-recall curve.

It can be seen from the precision-recall curve in Figure 8 that our model performs better than the basic model in the Bot-IoT-10 scenario. Due to the lack of valid bytes that can be recognized by classification models, both models perform poorly in Service-scan and Os-scan traffic, but our model performances also significantly better on service-scan and keylogging and the AP of our model are higher ,21% and 24%, respectively, than that of the basic model. A reasonable explanation is that, as shown in Figure 7, the theft and OS-scan traffic lack effective bytes that the classification model can recognize. Compared with the channel attention mechanism, the coordinate attention mechanism can make our model focus and distribute weight more on effective fields of short traffic to improve the performance of the model in identifying traffic with fewer effective bytes, which is very beneficial to the classification of traffic. And the multiscale feature fusion module uses different convolution kernel to parallel multibranch extract traffic image features to enhance the recognition ability of the proposed model to traffic texture features and improve the classification performance in the categories where samples are scarce.

Finally, the accuracy of our model reaches 99.82% at the expense of 0.26 M parameters and 5.26 M FLOPs by embedding the CA module and replacing the multiscale feature fusion structure. Consequently, our model can classify traffic quickly and achieves both precise classification and lightweight.

*5.3. Effectiveness Analysis.* In this section, to validate the merits of our model, we compare our model with the following classic models: (1) 1D-CNN, (2) CNN-LSTM, (3) BLSTM, and (4) BGRUA model in all experimental scenarios in Section 4.3. The comparison of traffic classification models in four experimental scenarios are shown in Figure 9.

It can be seen that all five classification models can perfectly solve the classification problems in Encrypted-2 and Bot-IoT-3. Among the four experimental scenarios, the classification performance of 1D-CNN is the weakest because lacks the ability to extract interactive information, which leads to the fact that pure CNN often combines with RNN to extract the flow features. The experimental results of Encrypted-2 and Encrypted-20 show that our model in the encrypted traffic scenario has no difference from the convention and is also suitable for the encrypted traffic classification task. The experimental results of Bot-IoT-10 show that our model outperforms the baseline models on all metrics. The accuracy, precision, recall, and F1-score rates reached 99.82%, 99.6%, 99.5%, and 99.5%, respectively. Our model can still maintain an accuracy rate of 99.02% in more complex Encrypted-20 scenarios, which is 10.87%, 9.52%, 8.12% and 4.52% higher than that of the 1D-CNN model, CNN-LSTM model, BLSTM model, and BGRUA model, respectively.

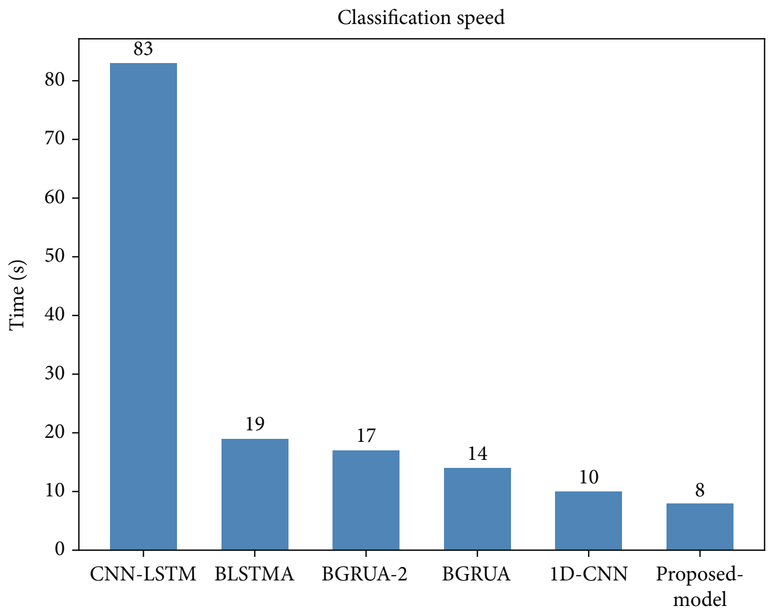


Figure 10: Actual deployment classification speed in Bot-IoT-10.

The performance of our model under the four scenarios has certain advantages for the following reasons: (1) CNN has unique advantages in packet-level features extraction because of local weight sharing. As the best lightweight CNN model, the reverse residual structure constructed by Mobilenetv3 solves the problem of gradient disappearance caused by network depth, making the shallow network can be trained by gradient, which increases the feature expression ability of the model. The normalization of Mobilenetv3 after the convolution layer also alleviates the gradient disappearance problem to a certain extent; (2) the improved attention mechanism in our model focuses on the more critical traffic byte location(flow-level features) in traffic classification and suppresses the characteristics of unimportant traffic byte location; (3) the linear bottleneck structure without reverse residual in the basic model is replaced by multiscale feature fusion structure, which takes into account the extraction of traffic texture semantic features and geometric details, enhancing the feature extraction ability of traffic.

*5.4. Efficiency Analysis.* First, we evaluate the efficiency of our model in terms of the model parameters and computation complexity; we recorded each traffic classification model parameters and Flops in Bot-IoT-10 scenario.

As shown in Table 7, our model benefits greatly from our improvement above. The number of parameters of our model is 4.45% of 1D-CNN, 10.87% of CNN-LSTM. Concerning computation, our model is 0.26% of 1D-CNN, 0.07% of CNN-LSTM, and 28.18% of the BLSTM model. Compared with the lightweight model BGRUA, although our model parameters are slightly more, the computation has been attenuated by 62.64%. This is because the dependencies between units in BGRUA make it time-consuming to wait for the output of the previous unit. Mobilenetv3 is carefully designed to reduce the model parameters and computation amount as much as possible from many aspects, such as replacing the full connection layer with two PWs, using a large number of linear bottleneck layers with deep separable convolutions, and replacing the activating function Relu with low-consumption h-swish. These designs effectively reduce the parameters and computation and accelerate the speed of traffic classification to a certain extent, making this model competent for the deployment and application of edge devices facing the shortage of hardware resources.

Second, to verify the capability of fast traffic classification for edge devices with limited resources, we deploy our model, BGRUA, BLSTMA, and CNN-LSTM on the classic edge device Raspberry Pi 4. At the same time, to fully simulate the device with limited resources, we lock three cores of the Raspberry CPU, only use a single core to test. In order to make a fair comparison, we randomly select 100 traffic flows of Bot-IoT and use each model to continuously identify the selected traffic flows in Raspberry Pi 4 in Bot-IoT-10 scenario. At the same time, we record the time taken to classify 100 flows, and the average classification speed is shown in Figure 10.

From Figure 10, it can be seen that our model only average takes 8 ms for a single flow classification on a single-core Raspberry Pi 4 Model B, much faster than other traffic classification models, which reflects our model has the best real-time effect in the limited resource. It is worth noting that 1D-CNN with high computational complexity is significantly faster than BGRUA and LSTM unexpectedly, this is because both of them use serial calculation in the operation process. The computation of the hidden state at the current time not only depends on the input at the current time, but also depends on the output at the previous time. The calculation at the current time can only be carried out after the calculation at the previous time is completed.

This dependency between various dimensions in the front and back time steps makes both LSTM and GRU much slower than other models in actual deployment. In summary, our model is a lightweight deep learning model for efficient classification traffic of edge devices.

*5.5. Remarks.* In this section, we compare our model with the other four baseline models in terms of effectiveness and efficiency.

For effectiveness, the performance of our model under the four scenarios has certain advantages. According to the effectiveness analysis in the Encrypted-20 scenarios, 10.87%, 9.52%, 8.12%, and 4.52% higher than that of the 1D-CNN model, CNN-LSTM model, BLSTM model, and BGRUA model, respectively, which demonstrate that our model performs well for traffic classification.

For efficiency, our model achieves the optimal in terms of the number of parameters, the computation, and testing speed. More specifically, the number of parameters of our model is 4.45% of 1D-CNN, 10.87% of CNN with LSTM, 136.84% of the BLSTM, and 185.71% BGRUA model, and the computation of our model is 0.26% of 1D-CNN, 0.26% of CNN-LSTM, 28.18% of BLSTM, and 62.64% of BGRUA model. At last, the deployment test on Raspberry Pi checks the superiority of our model in the edge device with limited resources. In short, our model is excellent in terms of effectiveness and efficiency.

## 6. Conclusions

Burning requirements on the edge traffic classification are put forward due to the widespread adoption of wireless communications and mobile computing. This paper proposes a lightweight model based on Mobilenetv3, which compresses the scale, width, and resolution to reduce the model weight and computation consumption, embeds the coordinate attention mechanism and the multiscale feature fusion to ensure the perfect balance between classification performance and lightweight. The experimental results show our model has the smallest number of parameters and the highest computing efficiency while ensuring high classification accuracy. Compared with the lightweight BGRUA traffic classification model, although the number of parameters is similar, the accuracy is increased partly, and the computation amount is greatly reduced by 62.64%. In addition, the actual deployment test on Raspberry Pi proves that the model in this paper can achieve high-accuracy traffic classification at a low cost and is suitable for traffic classification in edge devices with limited computing resources. Future research will focus on deploying traffic classification models in a wide range of edge networks using federal semisupervised learning.

## Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declared that they have no conflicts of interest regarding this work.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62176264).

## References

[1] A. Headquarters, "WAN and Application Optimization Solution Guide Cisco Validated Design," in 2008.

[2] M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: a survey," *Sustainable Cities and Society*, vol. 60, article 102177, 2020.

[3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communication Surveys and Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[4] "Snort—the de facto standard for intrusion detection/prevention," 2007, http://www.snort.org.

[5] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[6] M. S. Sheikh and Y. Peng, "Procedures, criteria, and machine learning techniques for network traffic classification: a survey," *IEEE Access*, vol. 10, pp. 61135–61158, 2022.

[7] N. Ivanov, "Unleashing the Internet of things with in-memory computing—IoT now—how to run an IoT enabled business," 2019, https://www.IoT-now.com/2019/01/17/92200-unleashing-internet-things-memory-computing.

[8] A. Hameed, J. Violos, and A. Leivadeas, "A deep learning approach for IoT traffic multi-classification in a smart-city scenario," *IEEE Access*, vol. 10, pp. 21193–21210, 2022.

[9] M. Z. F. Audah, T. S. Chin, Y. Zulfadzli, C. K. Lee, and K. RizaluddinI, *Towards Efficient and Scalable Machine Learning-Based QoS Traffic Classification in Software-Defined Network*, Springer International Publishing, 2019.

[10] W. Niu, Z. Zhuo, X. Zhang, X. Du, G. Yang, and M. Guizani, "A heuristic statistical testing based approach for encrypted network traffic identification," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3843–3853, 2019.

[11] A. Pektaş and T. Acarman, *Identification of Application in Encrypted Traffic by Using Machine Learning*, Springer International Publishing, 2017.

[12] J. Cheng, Y. Wu, E. Yuepeng et al., "MATEC: a lightweight neural network for online encrypted traffic classification," *Computer Networks*, vol. 199, p. 108472, 2021.

[13] A. Moore, D. Zuev, and M. Crogan, *Discriminators for Use in Flow-Based Classification*, Queen Mary University of London, London, 2013, https://www.cl.cam.ac.uk/~awm22/publications/moore2005discriminators.pdf.

[14] C. Liu, Z. Cao, G. Xiong, G. Gou, S. M. Yiu, and L. He, "MaMPF: encrypted traffic classification based on multi-attribute Markov probability fingerprints," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Banff, AB, Canada, 2018.

[15] Y. Okada, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka, "Comparisons of machine learning algorithms for application identification of encrypted traffic," in *2011 10th International Conference on Machine Learning and Applications and Workshops*, pp. 358–361, Honolulu, HI, USA, 2011.

[16] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43–48, Beijing, China, 2017.

[17] T. Shapira and Y. Shavitt, "FlowPic: a generic representation for encrypted traffic classification and applications identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1218–1232, 2021.

[18] Y. Wang, H. Zhou, and H. Feng, "Network traffic classification method basing on CNN," *Journal on Communications*, vol. 39, no. 1, pp. 14–23, 2018.

[19] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[20] Z. Zou, J. Ge, H. Zheng, Y. Wu, C. Han, and Z. Yao, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 329–334, Exeter, UK, 2018.

[21] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, article 106944, 2019.

[22] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameter and <0.5MB model size," 2016, http://arxiv.org/abs/1602.07360.

[24] S. Roy, T. Shapira, and Y. Shavitt, "Fast and lean encrypted Internet traffic classification," *Computer Communications*, vol. 186, pp. 166–173, 2022.

[25] K. Fauvel, A. Finamore, L. Yang, F. Chen, and D. Rossi, "A lightweight, efficient and explainable-by-design convolutional neural network for internet traffic classification," 2022, http://arxiv.org/abs/2202.05535.

[26] X. Liu, J. You, Y. Wu et al., "Attention-based bidirectional GRU networks for efficient HTTPS traffic classification," *Information Sciences*, vol. 541, pp. 297–315, 2020.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, http://arxiv.org/abs/1406.1078.

[28] S. Zhang, Y. Bu, B. Chen, C. Sun, H. Wang, and X. Hu, "Encrypted traffic classification based on multi-layer bidirectional sru and attention model," *Computer Engineering*, vol. 38, no. 1, pp. 1–15, 2022.

[29] K. Kim, J. H. Lee, H. K. Lim, S. W. Oh, and Y. H. Han, "Deep RNN-based network traffic classification scheme in edge computing system," *Computer Science and Information Systems*, vol. 19, no. 1, pp. 165–184, 2022.

[30] A. Howard, M. Sandler, G. Chu et al., "Searching for Mobilenetv3," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2020.

[31] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 13713–13722, 2021.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[33] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[34] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International conference on information networking (ICOIN)*, pp. 712–717, Da Nang, Vietnam, 2017.