Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.036701
Article

MTC: A Multi-Task Model for Encrypted Network Traffic Classification Based on Transformer and 1D-CNN

Kaiyue Wang[1], Jian Gao[1,2,*] and Xinyan Lei[1]

[1]Department of Information Network Security, People's Public Security University of China, Beijing, 102600, China
[2]Safety Precaution Laboratory of Ministry of Public Security, Beijing, 102600, China
*Corresponding Author: Jian Gao. Email: gaojian@ppsuc.edu.cn
Received: 09 October 2022; Accepted: 22 December 2022

Abstract: Traffic characterization (e.g., chat, video) and application identification (e.g., FTP, Facebook) are two of the more crucial jobs in encrypted network traffic classification. These two activities are typically carried out separately by existing systems using separate models, significantly adding to the difficulty of network administration. Convolutional Neural Network (CNN) and Transformer are deep learning-based approaches for network traffic classification. CNN is good at extracting local features while ignoring long-distance information from the network traffic sequence, and Transformer can capture long-distance feature dependencies while ignoring local details. Based on these characteristics, a multi-task learning model that combines Transformer and 1D-CNN for encrypted traffic classification is proposed (MTC). In order to make up for the Transformer's lack of local detail feature extraction capability and the 1D-CNN's shortcoming of ignoring long-distance correlation information when processing traffic sequences, the model uses a parallel structure to fuse the features generated by the Transformer block and the 1D-CNN block with each other using a feature fusion block. This structure improved the representation of traffic features by both blocks and allows the model to perform well with both long and short length sequences. The model simultaneously handles multiple tasks, which lowers the cost of training. Experiments reveal that on the ISCX VPN-nonVPN dataset, the model achieves an average F1 score of 98.25% and an average recall of 98.30% for the task of identifying applications, and an average F1 score of 97.94%, and an average recall of 97.54% for the task of traffic characterization. When advanced models on the same dataset are chosen for comparison, the model produces the best results. To prove the generalization, we applied MTC to CICIDS2017 dataset, and our model also achieved good results.

Keywords: Encrypted traffic classification; multi-task learning; feature fusion; transformer; 1D-CNN

1 Introduction

Experts and academics have been drawn to research network traffic characteristics and classification techniques in past years due to the rising demand for high-speed network transmission. Global Internet traffic will reach 396 EB per month in 2022, according to the Cisco Visual Networking Index 2017–2022, [1]. This indicates that effective network management and the performance of services like network survivability, differential quality of service (QoS) assurances, and dynamic access control can both be significantly enhanced by adequate network traffic processing, [2]. As technology evolves, encrypted traffic is becoming a heavier portion of all traffic, which means that research into the classification of encrypted traffic is urgent.

There are different classification tasks for network traffic classification in different application scenarios, such as attack traffic detection, [3], video traffic classification, [4], etc. Application identification and traffic characterization are two of the more significant duties. Application identification refers to identifying traffic based on the particular application that generated that traffic, such as FTP, Facebook, etc., whereas traffic characterization refers to classifying traffic based on protocol families, such as chat, voice, etc. For these two duties independently, the majority of current solutions adopt a single-task approach, adding to the complexity of network management.

It has become more challenging for traditional port-based and payload-based methods to achieve good results as time has gone on, [5], and some applications have started to use non-standard ports for transmission. Additionally, because traditional machine learning techniques require a lot of labor to extract features, deep learning-based traffic classification methods are more efficient.

CNN and Transformer [6] has demonstrated improved performance in the wide area of artificial intelligence, including computer vision [7–9], natural language processing [10], as well as traffic classification in recent years. Compared to 2D-CNN, 1D-CNN is more suited to traffic classification tasks, because traffic may be thought of as sequential data, [11]. However, CNN has the issue of a small receptive field because of the computational properties of the convolution operator, which makes it challenging to find long-distance correlation information when working with extended sequences. With better outcomes, Transformer has also been used in the field of traffic classification, [12,13]. Studies have revealed, however, that Transformer processes lengthy sequences with excellent efficiency while processing short sequences does not significantly improve and it struggles to uncover detailed features while processing short sequences, [14].

As we can see from the above, both Transformer and 1D-CNN have weaknesses when it comes to handling traffic sequences, and we have reason to believe that the lack of local detailed or global representation of traffic features will reduce the effectiveness of the classifier. However, if we combine their strengths, we can make them work together to their full potential. Therefore, we assume that Transformer and 1D-CNN have some complementarity. The feature information obtained by 1D-CNN can optimize the local detailed features missing from Transformer, while the feature information obtained by Transformer can make up for the long-distance correlation information lacking by 1D-CNN, enhancing the feature representation of traffic sequences. In the model proposed in this paper, the features of both are combined by a parallel network structure, and on ISCX VPN-nonVON dataset, application identification and traffic characterization are carried out concurrently using multi-task learning, which enhances traffic classification performance. And on another dataset CICIDS2017, MTC performed both the attack type classification and malicious traffic identification tasks, also achieving excellent results.

The main contributions of this paper are as follows:

(1) In order to extract both local detail features and acquire more long-distant correlation information in the network traffic, a new classification model for encrypted traffic is proposed. It combines Transformer and 1D-CNN to classify encrypted network traffic, uses a parallel structure to learn multiple tasks simultaneously and improves the classification effect.
(2) On the benign traffic dataset encrypted using VPN and Tor, ISCX VPN-nonVPN, and the malicious traffic dataset encrypted by SSH and HTTPS, CICIDS2017, the average F1 score and recall attained by MTC exceeded those of the multi-task Transformer, the multi-task 1D-CNN model and other state-of-the-art methods, showing good performance and generalization of MTC.
(3) We designed a feature fusion block to fuse the features generated by Transformer and 1D-CNN blocks with one another. This block compensates for the shortcomings of Transformer in dealing with short sequences and 1D-CNN in dealing with long sequences, allowing our model to perform well on both long or short sequence datasets, indicating that our model has great potential for application.

2 Related Works

Network traffic classification is a multi-classification problem, which refers to the classification and identification of the types to which a network traffic packet or flow belongs according to certain characteristics. Port-based, payload-based and machine-learning methods are the most often used technical methods for traffic classification. The first two approaches are no longer effective due to development of network encryption technology, and the prevailing current methods are built using conventional machine learning and deep learning methods. The motivation for the research in this paper will be derived from the related work in the area of traffic classification.

Traditional machine learning methods have achieved good results in traffic classification. A network model based on Naive Bayes is suggested to identify video traffic in real-time, [4], which enhances the covariance matrix of characteristics and transforms it into a positively defined symmetric matrix to improve classification accuracy. Reference [15] suggests a simple methodology that uses the TCP protocol and the K-means clustering technique to identify the application traffic and achieves superior results. Although machine learning techniques have some advantages, both models are limited to video traffic or TCP applications and have poor generalization capabilities. They also largely rely on manually extracted features.

Deep learning techniques are widely utilized in the field of traffic classification as a solution to the issues mentioned, and can be used to get greater outcomes while spending less on labor. CNN networks stand out among them in terms of traffic classification. Reference [11] proposes an end-to-end traffic classification approach and experimentally shows that 1D-CNN outperforms 2D-CNN in traffic classification tasks because 1D-CNN is better suited to sequential or language input, whereas traffic is fundamentally a sequence of bytes. A similar CNN-based deep neural network model was made, and the PCA approach was used, which significantly reduced memory cost and improved performance, [16]. Another method using CNN methods is to turn raw data into grayscale maps. Reference [17] slices the raw traffic data into flows and transforms them into image format for classification using a 2D-CNN-based model. This method increases classification accuracy but is more expensive because it consumes a lot of time and memory during the data preprocessing stage.

With the advent of Transformer, traffic classification jobs began to see an increase in the number of methods based on the attention mechanism and Transformer. Reference [18] proposes a lightweight

model that lowers the training cost while increasing accuracy by fusing an attention mechanism with a long short-term memory network (LSTM). An ET-BERT network is proposed and is based on the Transformer-based natural language processing pre-training model, Bert, [10]. This network increases classification accuracy on numerous datasets and exhibits high generalization, [12]. All these models mentioned above employ the single-task model, while multi-task learning also performs better in the area of traffic classification.

In contrast to single-task models, multi-task learning enables the simultaneous training of multiple tasks, which reduces the training cost. Since the multiple tasks are not independent of one another and the learning of one task can improve the results of another, it is possible to simultaneously produce better results for multiple tasks, [19]. A proposed 1D-CNN-based multi-task traffic classification model allows for the simultaneous performance of three tasks, bandwidth demand, flow length, and traffic type classification, while reducing the need for labeled data [19]. MTT is a multi-task traffic classification model based on Transformer, which eliminates a significant number of training parameters and increases classification accuracy compared to the 1D-CNN model, [13]. The experiments in this reference demonstrate that when the input data length is changed from 1500 bytes to 150 and 100 bytes, the F1 score obtained by Transformer decrease by 0.58% while the F1 score obtained by 1D-CNN increase by 0.62%, illustrating the benefits and drawbacks of these two models in handling long and short traffic sequences. The above models all use 1D-CNN or Transformer alone, but as we have discussed, 1D-CNN and Transformer have some disadvantages when dealing with sequences of different lengths. This is because the 1D-CNN has a smaller receptive field and is more suitable for extracting local features, while the Transformer is the opposite, so this point leads to our research motivation. We propose a conjecture that if we can fuse the local features extracted by 1D-CNN and the global extracted by Transformer, then we can compensate for the disadvantages of both, so that the model can be trained with both local and global features, and the results must be better than using 1D-CNN and Transformer alone, which has been proved by experiments. Based on the foregoing research, this work offers a multi-task encrypted traffic classification model, MTC, with a parallel structure that combines the features produced by Transformer and 1D-CNN to address each method's processing limitations and increase classification performance.

3 Methodology

3.1 Dataset

3.1.1 ISCX VPN-nonVPN

In this paper, we analyze the model performance using the publicly available dataset ISCX VPN-nonVPN. The original dataset was split into 14 categories, including 7 categories for ordinary traffic and 7 categories for traffic that was encrypted using VPN techniques including chat, VPN-chat, browsing, VPN-browsing, etc. The dataset covers 17 different application traffic, [20]. This dataset is excellent for testing the traffic classification model in this paper because it is real rather than produced manually and has a wide variety of data types. Given that the labels "Browsing" and "VPN-Browsing" could be mistaken for others, we follow other studies' methodology, [11,13,18,21,22], and eliminate this label before reclassifying the dataset into 12 traffic type labels and 17 application labels, as shown in Table 1. Reference [11] provides information about the traffic type labels comprised of the application traffic.

Table 1: Traffic type labels, application labels, and auxiliary task labels

Traffic type labels	Application labels	Auxiliary task labels
Chat, Email, File Transfer, Streaming, Torrent, Voip, VPN: Chat, VPN: Email, VPN: File Transfer, VPN: Streaming, VPN: Torrent, VPN: Voip	AIM Chat, Email, Facebook, FTPS, ICQ, Gmail, Hangouts, Netflix, SCP, SFTP, Skype, Spotify, Torrent, Tor, Vimeo, Voipbuster, Youtube	All_Chat, All_Email, All_File Transfer, All_Streaming, All_Torrent, All_Voip

3.1.2 CICIDS2017

To validate the generalization and application potential of our proposed model, we apply the publicly encrypted malicious traffic dataset CICIDS2017 to the experiments. This dataset contains benign and multiple types of common attack traffic, and consists of 78 figures representing extracted traffic features, which are short sequences compared to VPN-nonVPN, so these two datasets were also selected to be used in our experiments to verify the model's ability to handle long or short sequences.

3.2 Data Pre-processing

3.2.1 Pre-processing

The input forms of traffic can be broadly categorized into four groups in the traffic classification model: flow-based, [23], session-based, [11], packet-based, [13,21], and mixed input forms, [24]. Typically, flow-based, session-based, and mixed input forms contain more feature information. However, to decrease the data pre-processing burden, as well as computational and storage resources, we select a single packet as the input to our model on ISCX VPN-nonVPN dataset.

For VPN-nonVPN, the original packets contain some redundant information, such as MAC addresses and IP addresses, that affects the outcomes of traffic classification, the data must be pre-processed, as illustrated in Fig. 1.

(1) The Ethernet header with the physical layer information was first eliminated because it is useless for traffic classification and will cause the model to overfit.
(2) Second, the IP address in the IP header was changed to 0.0.0.0 for the same reason.
(3) Third, the size of the transport layer protocol headers was standardized by reducing the size of the TCP header to 20 bytes and padding the UDP header, which is typically 8 bytes, to 20 bytes with 0 bytes.
(4) Additionally, we eliminated the packets used mostly for service protocols like DNS, ACK, SYN, and FIN packets, which implement the three TCP handshakes and don't include any actual information.
(5) Finally, since the input to the model requires the same size data, each packet is then padded at the end with 0 bytes up to 1500 bytes, and all bytes are divided by 255 to normalization. This 1500-byte traffic sequence is the initial input data format.

For CICIDS2017, we removed ten features with less difference and 17 features with redundant information, and cleaned outliers like NaN values, and finally used the quantile transformation method to normalization. Therefore, the final data is a sequence containing 49 figures of [0–1]. Due to length issues of paper, we do not show the names of the 49 features that were specifically preserved. Reference [25] shows the detailed description of pre-processing process on this dataset.

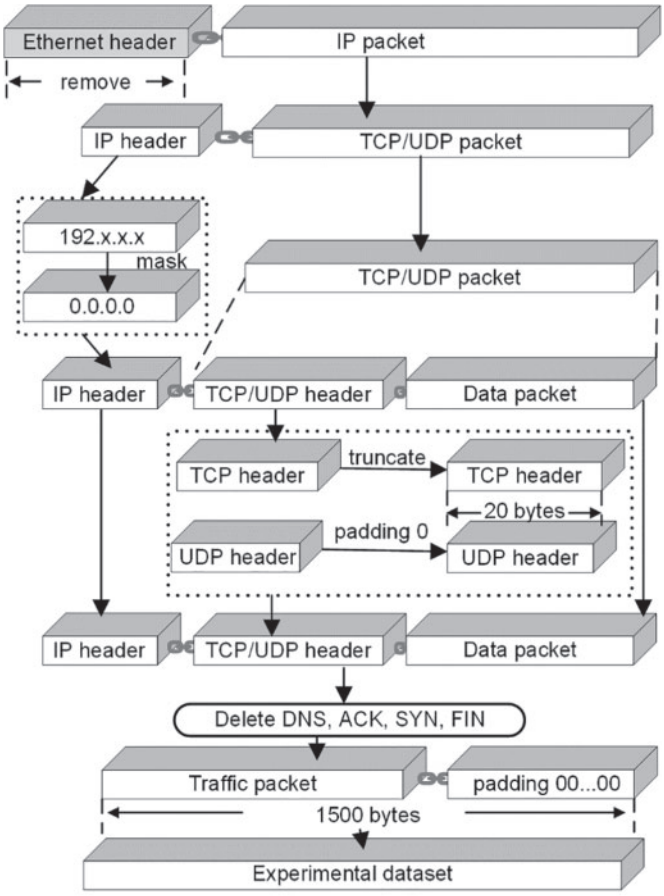

Figure 1: Data preprocessing

3.2.2 Labeling

Since our model is a multi-task learning model and each input dataset has several labels, the following step is to label the data. According to Table 1, we will first categorize the VPN-nonVPN data with application labels (there are a total of 17 categories) and traffic type labels (there are a total of 12 categories). Based on the hard parameter sharing property of the multitask model, a simpler auxiliary task that is closely related to the complex task can improve the classification accuracy of the complex task while performing the complex classification task, [26], therefore, we introduce an auxiliary task, which does not distinguish between encrypted and regular traffic but only classifies them by type, i.e., reducing the 12 labels to 6 labels, [13], as shown in Table 1.

On the dataset CICIDS2017, we designed two classification tasks that attack type classification (Attack.) which means classifying multiple attack types contained in malicious samples from all samples including benign and malicious, and malicious traffic identification (Mal.) which is a binary classification that refers to identifying whether a certain sample is benign or malicious from the entire dataset. That is, we tag this dataset with two types of labels, an Attack. label (7 categories before balancing) and a Mal. label (2 categories), as shown in Table 2.

Table 2: Class distribution of CICIDS2017 dataset

Attack. label	Mal. label	Imbalanced	Balanced
Benign	Benign	2035505	1330000
DoS/DDoS		320269	350000
PortScan		57305	75500
Brute force	Malicious	8551	66000
Web attack		2118	65500
Botnet ARES		1943	65000
Infiltration		36	

3.2.3 Processing Class Imbalance

Additionally, Fig. 2 displays the histogram of the sample number of packets for 12 different traffic types and 17 different application traffic types. It is clear from these graphs that there is a serious class imbalance issue in ISCX VPN-nonVPN, which, if used directly, will negatively impact the classification performance of the model. In order to tackle this issue, 50,000 samples from each of the 17 types of application traffic packets—selected equally from the ordinary and encrypted traffic—are taken using random sampling. If the total number of samples is less than 50,000, all samples are taken to achieve relative balance.

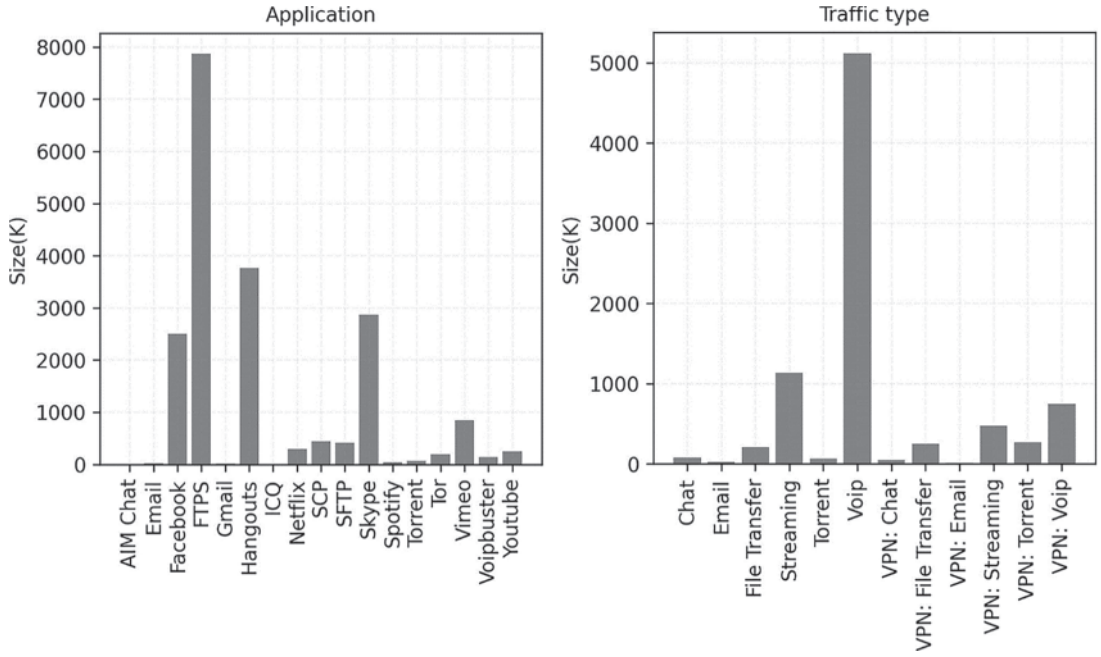

Figure 2: The number of sample packets for application traffic and traffic type

As shown in Table 2, the class imbalance problem is more severe at CICIDS2017, for example, the Infiltration class sample only accounts for approximately 0.00001 of the total sample size. We therefore removed the infiltration class samples from the data and resampled the samples from other

classes using the RandomUnderSampler and oversampling SMOTE techniques, the balanced data distribution are presented in Table 2 and the final Attack. labels have six classes.

3.3 Architecture

3.3.1 Overview

We developed a parallel multi-task encrypted network traffic classification model, MTC, the general architecture of which is depicted in Fig. 3, to improve 1D-CNN's capacity to extract long-distance correlation information, make up for Transformer's weakness in local feature extraction when processing traffic sequences, and achieve the purpose of complementing each other's strengths and weakness.

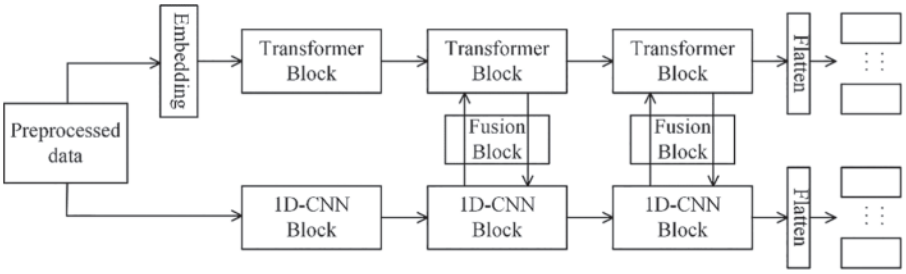

Figure 3: The overall framework of the model

The MTC comprises a feature fusion block, a classification layer, a Transformer branch, and a 1D-CNN branch, each of which has three small blocks called Transformer Blocks and 1D-CNN Blocks. The pre-processed data is first input to the Transformer branch and the 1D-CNN branch. Next, the features produced by the Transformer branch are input to the CNN branch through the feature fusion block to allow the CNN branch to obtain more long-distance correlation information, and similarly, the features produced by the CNN are also input to the Transformer through to enhance the Transformer branch's ability to perceive local detailed features, so that the two branches can complement each other's extracted features to achieve the effect of complementing each other's strengths and weaknesses, thus optimizing the feature representation of encrypted traffic sequences and improving the classification performance. Due to the lack of a feature output in the first block, feature fusion is not done; nevertheless, it is conducted in the second and third blocks. The classification layer is reached after receiving the output of the third blocks, and classifies them using the softmax function.

3.3.2 Transformer Block

The Transformer block is shown in Fig. 4a, which is based on the encoder part of Transformer, [6]. It is important to alter the dimensionality of the input data before putting it into the Transformer block since the data input format for the Transformer branch is different from the one for the 1D-CNN branch. Using dataset VPN-nonVPN as an example, the Embedding function converts the 1500-byte traffic sequence into n d-dimensional tokens, where $n \times d$ equals 1500. The 1D-CNN model already contains the location information of each "traffic token" in the traffic sequence, therefore this step does not involve extracting the positional information from the traffic sequence. Next, the data is merged with the local detail features extracted by the 1D-CNN block and normalized into an encoder with m repetitions, after which the data is fed into H multi-head attention layers, which are computed

as follows:

$$Q = ZW^Q; K = ZW^K; V = ZW^V \tag{1}$$

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_V}}\right)V \tag{2}$$

where $Z \in \mathbb{R}^{n\times d}$ represents the input to the encoder, $W^Q, W^K \in \mathbb{R}^{d\times d_K}$, $W^V \in \mathbb{R}^{d\times d_V}$ is the parameter learned by the self-attentive mechanism, set $d_K = d_V = d/H$. The multi-headed attention layer computes the self-attention outputs of the H heads simultaneously, integrating the results according to the following equation:

$$\text{MultiHead}\,(Z) = \text{Concat}\,(H_1, H_2, \ldots H_6)\,W^M \tag{3}$$

where $H_i, i \in 1, 2, \ldots 6$ is each self-attention output and $W^M \in \mathbb{R}^{Hd_V\times d}$ is the learned parameter, the result is input to the feedforward layer after residual connection and normalization, which consists of two fully connected layers, and the dropout function is added to each layer with the parameter set to 0.1, and the final output of the Transformer branch is obtained after residual connection and normalization once more. Table 3 displays the basic parameters involved in the Transformer branch. In the three layers of the overall model, the number of repetitions m of the Transformer block at each layer is 1, 5, and 6, respectively, and the parameters H is 6, n is 30, and d is 50. The output dimensions of the two fully connected layers are 1024 and 50, respectively.

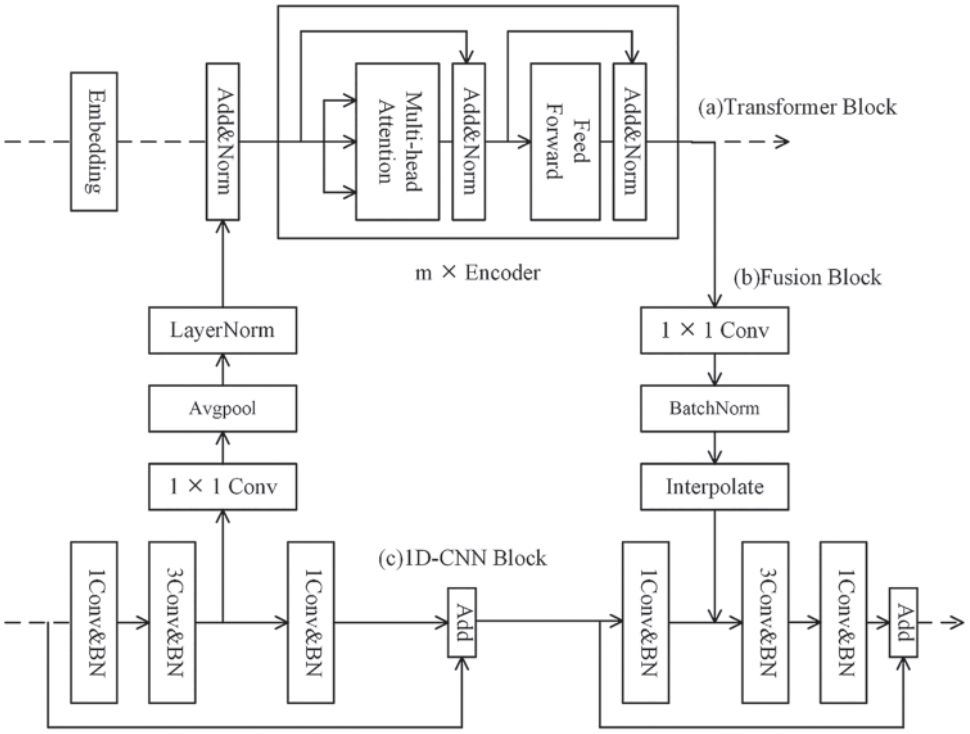

Figure 4: The specific structure of each branch module: (a) Transformer block; (b) Fusion block; (c) 1D-CNN block

Table 3: The basic parameters of each block in each layer on ISCX VPN-nonVPN

Layer	Transformer	Fusion block	1D-CNN
1	$\begin{bmatrix} H: 6, 30 \times 50 \\ 1 \times 1, 1024 \\ 1 \times 1, 50 \end{bmatrix} \times 1$		$\begin{bmatrix} 1 \times 1, 50 \\ 1 \times 3, 50 \\ 1 \times 1, 100 \end{bmatrix}$
2	$\begin{bmatrix} H: 6, 30 \times 50 \\ 1 \times 1, 1024 \\ 1 \times 1, 50 \end{bmatrix} \times 5$	$\leftarrow [1 \times 1, 50]$ $[1 \times 1, 50] \rightarrow$	$\begin{bmatrix} 1 \times 1, 50 \\ 1 \times 3, 50 \\ 1 \times 1, 100 \end{bmatrix}$ $\begin{bmatrix} 1 \times 1, 50 \\ 1 \times 3, 50 \\ 1 \times 1, 100 \end{bmatrix}$
3	$\begin{bmatrix} H: 6, 30 \times 50 \\ 1 \times 1, 1024 \\ 1 \times 1, 50 \end{bmatrix} \times 6$	$\leftarrow [1 \times 1, 50]$ $[1 \times 1, 100] \rightarrow$	$\begin{bmatrix} 1 \times 1, 100 \\ 1 \times 3, 100 \\ 1 \times 1, 200 \end{bmatrix}$ $\begin{bmatrix} 1 \times 1, 100 \\ 1 \times 3, 100 \\ 1 \times 1, 200 \end{bmatrix}$

3.3.3 1D-CNN Block

As seen in Fig. 4c, the 1D-CNN block is made up of one or two bottleneck convolutional structures, which are employed to lessen the computational load and the number of parameters, hence increasing the depth of the model layers, [27]. Each bottleneck consists of three convolutional layers of 1×1, 1×3, and 1×1, where each convolutional layer has a stride size of 1. A residual connection follows every bottleneck structure. Because feature fusion is not necessary yet, the initial 1D-CNN block is the only one with a bottleneck structure. There are two bottlenecks in the second and third 1D-CNN blocks. The local detail features of the traffic sequence produced by the 1D-CNN are fed to the feature fusion block after the first bottleneck block's 1×3 convolutional layer, and the long-distance correlation features produced by the Transformer block are received after the second bottleneck block's first 1×1 convolutional layer, after which the traffic sequence features of the 1D-CNN block are obtained. Table 3 shows the output dimension of the bottleneck structure for each 1D-CNN block. To avoid overfitting, a dropout layer is applied after each convolutional layer with a parameter value of 0.07.

3.3.4 Feature Fusion Block

The feature fusion block must be used to communicate the features produced by the aforementioned two blocks to fully fuse the local detail features of the traffic sequences produced by the 1D-CNN block and the Transformer block with the long-distance correlation information features. It is required to match the dimensionality of the traffic sequence feature information supplied by

both sides because the aforementioned two blocks output different feature dimensions, making it impossible for the features of the two blocks to receive and fuse with one another. The features from the 1D-CNN block are down-sampled using the average pooling function, while the features from the Transformer block are up-sampled using the interpolation function to achieve the goal of aligning the dimensionality, as shown in Fig. 4b. First, an 1×1 convolutional layer is required to raise or lower the dimensionality. Another normalization is required to complete the feature fusion process after the up-and-down sampling is finished. Table 3 displays the output dimensions of the feature fusion block's 1×1 convolutional layer.

3.3.5 Classification Layer

The classification layer of the multi-task learning model outputs multiple classification results. The MTC model has three outputs on VPN-nonVPN: traffic characterization (Tra.), application identification (App.) and auxiliary classification task (Aux.), and two outputs on CICIDS2017: attack type classification and malicious identification. Softmax and cross-entropy loss functions are used for classification. The final loss values are the weighted average of the loss values of each classification task by a certain ratio, where the App. task is based on the features generated by the Transformer branch for 17 classes, and the Tra. task and the Aux. task are based on the features generated by the 1D-CNN branch for 12 and 6 classes, respectively, and the weight ratio is 6:2:1. On CICIDS2017, the Attack. task is based on Transformer branch for 6 classes and Mal. Task is based on CNN branch for 2 classes, and the weight ratio is 2:1.

4 Experimental Results and Comparison

4.1 Experimental Setup

4.1.1 Experiment Environment

The experimental hardware environment for this experimental training and testing is Intel(R) Core(TM) i9-10980XE CPU@ 3.00 GHz, 18 cores, 36 threads, and 64 GB of RAM running on Ubuntu 20.04 LTS. The classification models were implemented using PyTorch 1.10 on NVIDIA RTX 3090 GPU card with 24 GB RAM, and using the CUDA 11.4 development toolkit with python 3.7.

4.1.2 Experiment Dataset

The encrypted traffic dataset ISCX VPN-nonVPN was used as the experimental dataset, and another dataset CICIDS2017 was also employed to validate the generalizability and to discuss the performance of the MTC when dealing with data of different lengths. The pre-processing process for these two datasets is detailed in Section 3.2. All experimental data was divided into training, validation, and test sets in the ratio of 64%, 16%, and 20%.

4.1.3 Model Parameters

On dataset VPN-nonVPN, the model is trained for a total of 100 epochs, with a batch size of 128, and the optimization Adam is applied to speed up the learning process, the initial learning rate is 0. 001, and the learning rate is updated dynamically using ReduceLROnPlateau with a factor of 0.1 and patience of 10, and regularization is performed using weight decay with a factor of 0.0001, and the model is prevented from overfitting using the early stopping technique with the patience of 20. The ReLU function is selected as the activation function.

4.2 Evaluation Metrics

For a better evaluation of the classification performance of the model, we use the F1 score (F1) and recall (Rc) as the model evaluation metrics. These metrics are calculated as shown below:

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4}$$

$$\mathrm{Rc} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5}$$

$$\mathrm{F1} = \frac{2\mathrm{R_c} \times \mathrm{P_r}}{\mathrm{R_c} + \mathrm{P_r}} \tag{6}$$

where Pr is the precision, and TP, FP, TN, and FN stand for the true positive, false positive, true negative, and false negative rates, respectively.

4.3 Comparison Models

Using the same dataset and two advanced models, two comparison models were created to compare with MTC. The first comparison model, called multi-task transformer, is to design a traffic classification model with multi-task learning using only Transformer based on the MTT, [13]. The second comparison model, referred to as multi-task 1D-CNN, is built on the deep packet, [21], with multi-task learning using just 1D-CNN. Results of comparison experiments demonstrate how MTC can fully merge the local detail features extracted by 1D-CNN and the long-distance correlation information features extracted by Transformer to have an improved classification performance of encrypted data. The experiment environment, experiment dataset, experiment parameters (Section 4.1.3), and evaluation metrics used in the comparison experimental model are all consistent with MTC.

4.3.1 Multi-Task Transformer Model

Fig. 5 depicts the organizational layout of the first comparison model, where input embedding fulfills the same role as the Embedding function in Section 3.3.2. Because the positional information of each byte in the encrypted traffic sequence is crucial for the Transformer model's classification, [13], this model differs from the Transformer branch of MTC in that positional encoding module must needs to be added before entering the multi-head attention layer. The calculation process for the positional encoding is specifically illustrated in the following equation.

$$\mathrm{PE}\,(pos, 2i) = \sin\left(pos \times 10000^{-\frac{2i}{d}}\right) \tag{7}$$

$$\mathrm{PE}\,(pos, 2i+1) = \cos\left(pos \times 10000^{-\frac{2i}{d}}\right) \tag{8}$$

After adding the positional encoding information, the data enters the encoder and performs the same operation as the encoder in Section 3.3.2, with an encoder repetition number of 7. In the classification layer, the multi-task Transformer model needs to output the classification results for multiple tasks simultaneously. Their respective loss weight ratios are 4:2:1 [13], on VPN-nonVPN, and 3:1 on CICIDS2017.

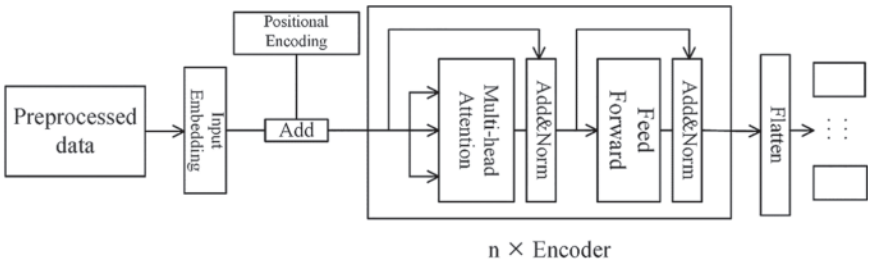

Figure 5: Multi-task transformer model

4.3.2 Multi-Task 1D-CNN Model

The specific structure of the multi-task 1D-CNN model is shown in Fig. 6, which contains two one-dimensional convolutional layers, a max pooling layer, and three full connection layers. The kernel size of the first convolutional layer is 4 and the stride is 3, the kernel size of the second convolutional layer is 5 and the stride is 1, the kernel size of the max pooling layer is 2 and the stride is 2, and the output dimensions of the three full connection layers are 200, 100, and 50, respectively. The BatchNorm is used for normalization and the dropout is added to the fully connected layer to prevent overfitting with a factor of 0.05. Finally, Softmax is still used to classify the multiple tasks with a loss weight ratio of 2:2:1 [13] on VPN-nonVPN and 3:1 on CICIDS2017.

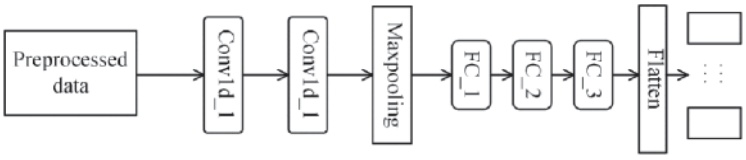

Figure 6: Multi-task 1D-CNN model

4.4 Results and Comparison

Table 4 compares the experimental outcomes of MTC developed in this paper with the advanced models Multi-task Transformer and Multi-task 1D-CNN using the same datasets ISCX VPN-nonVPN and CICIDS2017.

Table 4: Experimental results of F1 score (%) and recall rate (%) on MTC, multi-task transformer, multi-task 1D-CNN, and MTC-2task

Model	ISCX VPN-nonVPN						CICIDS2017			
	App.		Tra.		Aux.		Attack.		Mal.	
	F1	Rc	F1	Rc	F1	Rc	F1	Rc	F1	Rc
MTC	**98.25**	**98.30**	**97.94**	**97.54**	**97.97**	**97.81**	**99.47**	**99.66**	**99.71**	**99.75**
Transformer	97.27	97.18	95.90	95.44	96.86	96.50	99.30	99.52	99.66	99.70
1D-CNN	96.13	95.91	89.36	88.95	95.56	95.23	99.45	99.63	99.70	99.74
MTC-2task	95.60	95.24	92.46	91.53						

4.4.1 Results on ISCX VPN-nonVPN

Table 4 shows that the MTC outperforms the Multi-task Transformer and Multi-task 1D-CNN in terms of average F1 score and average recall for the three sub-tasks of application identification, traffic characterization, and auxiliary classification task. The average F1 score of the App. task is higher than the Multi-task Transformer and Multi-task 1D-CNN by 0.98% and 2.12%, respectively, and the average recall is 1.12% and 2.39% higher. For Tra. task, the average F1 is 2.04% and 8.58% higher than Multi-task Transformer and Multi-task 1D-CNN, respectively, and the average recall is 2.10% and 8.59% higher, respectively. The average F1 of Aux. task is 1.11% and 2.41% higher than Multi-task Transformer and Multi-task 1D-CNN, respectively, and the average recall is 1.31% and 2.58% higher, respectively. These results demonstrate that MTC can fully utilize the features of both Transformer and 1D-CNN blocks to extract long-distance correlation information and local detail information from the traffic sequences, which proves the conjecture we presented in Section 2. By doing so, MTC improves its ability to represent the features of the traffic sequences and thus outperforms Transformer and 1D-CNN in terms of performance.

Fig. 7 displays the line graphs of the detail experimental findings for each type of traffic, along with the F1 score and recall. MTC outperforms the results of the comparison models in terms of F1 score for each type of traffic in each subtask, except for All File Transfer in the Aux. task, and in terms of recall, except All_Streaming and All_Voip in the Aux. task, each of them having the best results. Additionally, when compared to the reference model, MTC outperforms it in traffic types with small sample numbers, such as AIM chat, ICQ, and chat, proving that our suggested model can be trained with unbalanced data and fewer labels.

4.4.2 Results on CICIDS2017

On the CICIDS2017 dataset, we replace the parameters n, d, H in the MTC model's Transformer block and Multi-task Transformer with 7, 7 and 7, respectively, because the data input length is different (from 1500 to 49). The other hyperparameters are unchanged. The experimental results of the attack type classification and malicious traffic identification tasks are shown in Table 4. As you can see, MTC achieves the best results, showing the good generalization. 1D-CNN, however, also achieves very excellent results, displaying its power in handling short sequences. Another point worth noting is that on the long sequence dataset, Transformer outperforms 1D-CNN, but when it comes to the short sequence dataset, 1D-CNN outperforms Transformer. Our model achieves optimal results for sequences of different lengths, again providing strong evidence that MTC can compensate for the shortcomings of Transformer and 1D-CNN and take full advantage of the strengths of both methods to improve performance in classification and identification tasks.

4.4.3 Comparison with Other State-of-the-Art Methods

In order to compare with the model MTC proposed in our paper, we selected several state-of-the-art methods on the same datasets in recent years, and the results are shown in Table 5. On the ISCX dataset, our model performs the best on both tasks. On the CICIDS2017, MTC performs the best on both tasks, except that the recall of malicious traffic identification is worse than LUCID [29]. For the recall of the Attack. task, we believe that the MTC model is also optimal in this item compared with DBN [25], because the recall of this task only provides the percentage value of one decimal point in the reference. These comparisons fully prove the progressiveness of MTC.

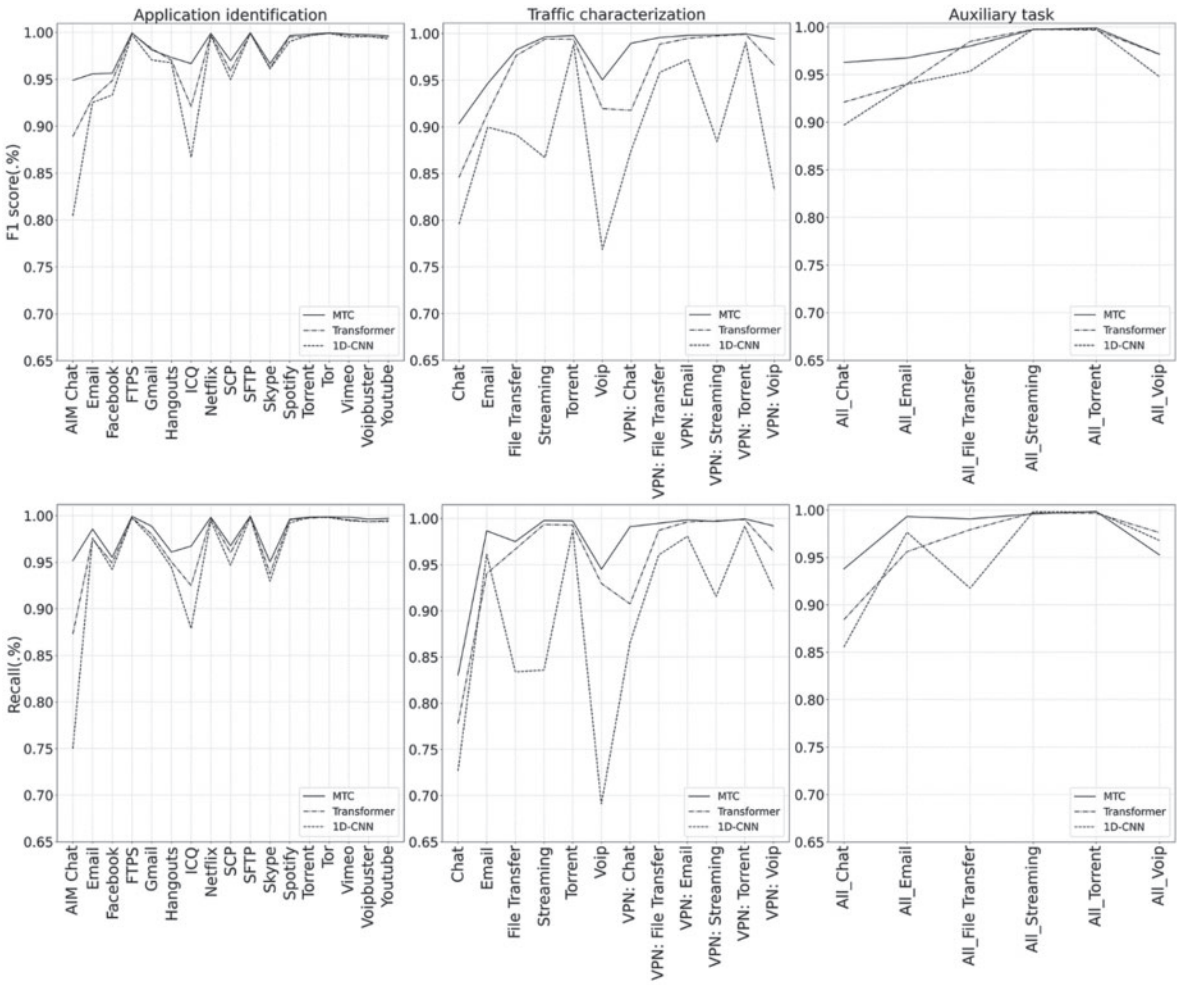

Figure 7: Various classes of traffic F1 score and recall including MTC, multi-task transformer and multi-task 1D-CNN on application identification, traffic characterization, and auxiliary classification

Table 5: Comparison with other state-of-the-art methods on ISCX VPN-nonVPN and CICIDS2017

Methods on ISCX VPN-nonVPN	App.		Tra.		Methods on CICIDS2017	Attack.		Mal.	
	F1 (%)	Rc (%)	F1 (%)	Rc (%)		F1 (%)	Rc (%)	F1 (%)	Rc (%)
1D-CNN [21], 2020	98.00	98.00	93.00	93.00	MLP [25], 2022	87.3	99.5	–	–
SAE [21], 2020	95.00	95.00	92.00	92.00	DBN [25], 2022	94.0	99.7	–	–
PERT [28], 2020	69.92	71.73	93.68	93.49	LUCID [29], 2020	–	–	99.66	**99.94**
TSCRNN [30], 2021	–	–	92.60	92.60	LSTM [31], 2019	–	–	93.97	89.89
CNN + GRU [32], 2020	96.08	–	–	–	CNN-LSTM [33], 2022	–	–	99.60	99.70
RBRN [34], 2020	97.21	97.33	–	–	AE + DNN [35], 2020	–	–	97.12	98.57
MTC	**98.25**	**98.30**	**97.94**	**97.54**	**MTC**	**99.47**	**99.66**	**99.71**	99.75

Note: The symbol "–" means that the task or metric was not provided in the reference.

5 Discussion

5.1 Hyperparameter Selection

The selection of hyperparameters has a significant impact on the outcomes of the experiments, and choosing the right ones can significantly boost the model's classification performance. This section primarily goes into detail about how we chose a few important hyperparameters on ISCX VPN-nonVPN.

5.1.1 Number of Repetitions of Encoder

We believe that the number of encoder repetitions is the key to the classification effect of the model, so we first determine the number of encoder repetitions in the second and third Transformer blocks of the model, according to the computational resources, we choose a total of four combinations in the experiments, as shown in Table 6, where m2, m3, represent the number of encoder repetitions in the second and third blocks of the model, respectively, the factor of dropout layers in the 1D-CNN block is 0.05, and the loss weight ratio of App., Tra. and Aux. are set to 4:2:1. And Table 6 shows that the model produces the best classification results when the number of encoder repetitions in the second Transformer block is 5 and in the third is 6.

Table 6: Experimental results of different numbers of encoders in the transformer block

m2, m3	App.		Tra.	
	F1 (%)	Rc (%)	F1 (%)	Rc (%)
5, 5	97.45	97.46	97.17	97.07
5, 6	97.75	97.68	97.73	97.26
6, 5	97.50	97.19	95.57	94.82
6, 6	96.70	96.48	95.46	96.17

5.1.2 Weight Ratio for Loss Function

When calculating the loss values of the model, the multi-task classification model must take a weighted average of the individual loss values of each subtask. Appropriately allocating the weights of each subtask's loss value plays a significant role in the classification performance of the model, and figuring out the weight ratio of the loss values is a crucial task for us, [36]. Table 7 displays the experimental results for various loss value weight ratios for the three subtasks. Due to the high correlation between the Tra. task and the Aux. task and the 2:1 ratio of the number of classifications for these two tasks, we fixed the loss weight ratios of the above two subtasks to 2:1 and changed only the loss value weights of the App. task. The results show that the outcomes are better when the loss weight ratio of the App, Tra, and Aux tasks is 4:2:1 and 6:2:1, and just one result of the F1 score of the Tra. task is 0.27% higher when the loss weight of the App. task is 4. So, the final loss value weight ratio we settle on is 6:2:1. The 1D-CNN blocks' dropout layer factor in the experiments is 0.05.

Table 7: Experimental results of different weight ratios of loss values for the three subtasks

Weight	App.		Tra.	
Ratio	F1 (%)	Rc (%)	F1 (%)	Rc (%)
3:2:1	93.28	91.93	92.04	91.79
4:2:1	97.75	97.68	97.73	97.26
5:2:1	95.71	95.53	92.38	91.44
6:2:1	97.82	97.78	97.46	97.48
7:2:1	97.70	97.53	97.26	96.95

5.1.3 Factor of Dropout Layers in 1D-CNN Blocks

The MTC model adds dropout layers after each convolutional layer of the 1D-CNN blocks in order to avoid overfitting and improve generalization, therefore we must choose the right dropout factor through experimentation. Table 8 displays the experimental findings. As can be seen, the model performs best for classification when the factor is set to 0.07, thus we decided to use that value for each dropout layer in the 1D-CNN blocks.

Table 8: Experimental results of different dropout parameter values in the 1D-CNN block

Dropout rates	App.		Tra.	
	F1 (%)	Rc (%)	F1 (%)	Rc (%)
0.05	97.82	97.78	97.46	97.48
0.07	98.25	98.30	97.94	97.54
0.09	94.69	94.43	88.18	89.56

5.1.4 Different Number of Epochs

The number of epochs is also important for the results of the model. If the number is too small, the model may not have time to fit the data, and if the number is too large, the problem of over fitting may occur, which will make the effect of the model on the testing set worse. We finally selected 100 epochs according to the results in Table 9.

Table 9: Experimental results of different number of epochs

Number of epochs	App.		Tra.	
	F1 (%)	Rc (%)	F1 (%)	Rc (%)
50	96.87	96.74	95.99	96.85
80	97.71	97.56	97.25	97.50
100	98.25	98.30	97.94	97.54
120	97.93	97.99	97.25	97.21

5.2 Impact of Auxiliary Classification Tasks

Finally, in order to verify the effect of auxiliary tasks on the classification performance of the model, we design a set of two-task models (MTC-2task), i.e., the model proposed in this paper is removed in the output layer of the 1D-CNN block to remove the auxiliary tasks for 6 classes of classification, keeping only the App. task of the Transformer block, and the Tra. task of the 1D-CNN block. Table 4 displays the experimental results, which clearly demonstrate how the auxiliary task can help improve classification performance. When the auxiliary task is removed, the model's classification performance declines significantly, as shown by the average F1 values of the App. task, which decrease by 2.65% and the average recall, which decrease by 3.06%; the average F1 value of the Tra. task, which decreases by 5.48% and the average recall, which decreases by 6.01%.

5.3 Limitations of This Work and Future Works

In the experiment, although the number of parameters is between Transformer and 1D-CNN, we feel that the training time of our model on the ISCX dataset is relatively long, which is the biggest shortcoming of our model. We guess that this is mainly because we use more encoder layers and the data length is long, which increases the training time undoubtedly. Next, we consider optimizing the model structure to reduce the training time on long sequences. The second disadvantage is that the interpretability of our model is still insufficient, which is known as a common issue with deep learning methods at the current stage. Although the experiments have yielded good results and proved our previous conjecture, it is difficult for us to describe how the model specifically features traffic data and represents the local and global features that be obtained by 1D-CNN and Transformer methods learning automatically in the potential space, which is also the focus of our future research work.

6 Conclusion

In this paper, we provide a Transformer and 1D-CNN-based multi-task encrypted traffic classification model. Multiple tasks are all simultaneously classified by the model using a parallel architecture. Through the feature fusion block, the model fuses the features produced by Transformer and 1D-CNN. This allows the model to fully utilize the local detail features and long-distance correlation information features produced by both methods, respectively, to improve the model's feature extraction capabilities and, as a result, the classification performance of encrypted traffic. The model delivers optimal results and can produce superior outcomes in the situation of data imbalance through comparison experiments with advanced ones on the two datasets, which shows good performance and generalization of our model.

Funding Statement: This work was supported by the People's Public Security University of China central basic scientific research business program (No. 2021JKF206).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] T. Barnett, S. Jain, U. Andra and T. Khurana, *Cisco visual networking index (vni) complete forecast update, 2017–2022*, 2018, [Online]. Available: https://oarklibrary.com/file/2/be52d44d-f201-475b-876d-6649efa14d85/e5235174-57e4-419c-a1bd-832c9fea6d07.pdf

[2] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura and A. Nucci, "Towards self adaptive network traffic classification," *Computer Communications*, vol. 56, pp. 35–46, 2015.

[3] Q. Dai, B. Zhang and S. Dong, "A DDoS-attack detection method oriented to the blockchain network layer," *Security and Communication Networks*, vol. 2022, 2022. https://www.hindawi.com/journals/scn/2022/5692820/
[4] K. L. Dias, M. A. Pongelupe, W. M. Caminhas and L. de Errico, "An innovative approach for real-time network traffic classification," *Computer Networks*, vol. 158, pp. 143–157, 2019.
[5] S. Sen, O. Spatscheck and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proc. of the 13th Int. Conf. on World Wide Web*, New York, NY, USA, pp. 512–521, 2004.
[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.,* "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[7] B. Li, Z. Sun, L. Tang, Y. Sun and J. Shi, "Detecting robust co-saliency with recurrent co-attention neural network," in *Int. Joint Conf. on Artificial Intelligence Organization (IJCAI-19)*, Macao, China, pp. 818–825, 2019.
[8] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang *et al.,* "Conformer: Local features coupling global representations for visual recognition," in *2021 IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, Montreal, QC, Canada, pp. 357–366, 2021.
[9] L. Tang, B. Li, S. Ding and M. Song, "Disentangled high quality salient object detection," in *2021 IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, Montreal, QC, Canada, pp. 3560–3570, 2021.
[10] J. Devlin, M. -W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
[11] 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 Int. Conf. on Intelligence and Security Informatics (ISI)*, Beijing, China, pp. 43–48, 2017.
[12] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi *et al.,* "ET-BERT: A contextualized datagram representation with Pre-training transformers for encrypted traffic classification," in *Proc. of the ACM Web Conf. 2022*, UK, pp. 633–642, 2022.
[13] W. Zheng, J. Zhong, Q. Zhang and G. Zhao, "MTT: An efficient model for encrypted network traffic classification using multi-task transformer," *Applied Intelligence*, vol. 52, pp. 10741–10756, 2022.
[14] H. Ren and Y. Wang, "Review of attention mechanism," *Journal of Computer Applications*, vol. 41, no. S1, pp. 1–6, 2021.
[15] Z. Iqbal, R. Rahim and I. A. Gillani, "Light-weight, real-time internet traffic classification," in *2019 IEEE Int. Conf. on Advanced Networks and Telecommunications Systems (ANTS)*, Goa, India, pp. 1–6, 2019.
[16] X. Zhang, G. Cheng and W. Zhang, "Network traffic classification method based on improved deep convolutional neural network," *SCIENTIA SINICA Informationis*, vol. 51, no. 1, pp. 56–74, 2021.
[17] T. Shapira and Y. Shavitt, "FlowPic: Encrypted internet traffic classification is as easy as image recognition," in *IEEE INFOCOM 2019—IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, pp. 680–687, 2019.
[18] W. Wei, H. Gu, W. Deng, Z. Xiao and X. Ren, "ABL-TC: A lightweight design for network traffic classification empowered by deep learning," *Neurocomputing*, vol. 489, pp. 333–344, 2022.
[19] S. Rezaei and X. Liu, "Multitask learning for network traffic classification," in *2020 29th Int. Conf. on Computer Communications and Networks (ICCCN)*, Honolulu, HI, USA, pp. 1–9, 2020.
[20] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proc. of the 2nd Int. Conf. on Information Systems Security and Privacy (ICISSP)*, Rome, Italy, pp. 407–414, 2016.
[21] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.
[22] M. Wang, K. Zheng, D. Luo, Y. Yang and X. Wang, "An encrypted traffic classification framework based on convolutional neural networks and stacked autoencoders," in *2020 IEEE 6th Int. Conf. on Computer and Communications (ICCC)*, Chengdu, Sichuan, China, pp. 634–641, 2020.

[23] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang *et al.,* "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Transactions on Big Data*, vol. 8, no. 1, pp. 241–252, 2019.
[24] S. Cui, B. Jiang, Z. Cai, Z. Lu, S. Liu *et al.,* "A session-packets-based encrypted traffic classification using capsule neural networks," in *2019 IEEE 21st Int. Conf. on High Performance Computing and Communications; IEEE 17th Int. Conf. on Smart City; IEEE 5th Int. Conf. on Data Science and Systems (HPCC/SmartCity/DSS)*, Zhangjiajie, Hunan, China, pp. 429–436, 2019.
[25] O. Belarbi, A. Khan, P. Carnelli and T. Spyridopoulos, "An intrusion detection system based on deep belief networks," in *Science of Cyber Security*, Matsue, Japan, Cham: Springer, pp. 377–392, 2022.
[26] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
[27] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, USA, pp. 770–778, 2016.
[28] H. He, Z. Yang and X. Chen, "PERT: Payload encoding representation from transformer for encrypted traffic classification," in *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*, Ha Noi, Vietnam, pp. 1–8, 2020.
[29] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.
[30] K. Lin, X. Xu and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT," *Computer Networks*, vol. 190, pp. 107974, 2021.
[31] M. Roopak, G. Tian and J. Chambers, "Deep learning models for cyber security in IoT networks," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conf. (CCWC)*, Las Vegas, NV, USA, pp. 0452–0457, 2019.
[32] C. Dong, C. Zhang, Z. Lu, B. Liu and B. Jiang, "CETAnalytics: Comprehensive effective traffic information analytics for encrypted traffic classification," *Computer Networks*, vol. 176, pp. 107258, 2020.
[33] A. Halbouni, T. Gunawan, M. Habaebi, M. Halbouni, M. Kartiwi *et al.,* "CNN-LSTM: Hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, pp. 99837–99849, 2022.
[34] W. Zheng, C. Gou, L. Yan and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *Proc. of the Web Conf. 2020*, New York, NY, USA, pp. 13–22, 2020.
[35] A. Bhardwaj, V. Mangat and R. Vig, "Hyperband tuned deep neural network with well posed stacked sparse autoencoder for detection of DDoS attacks in cloud," *IEEE Access*, vol. 8, pp. 181916–181929, 2020.
[36] A. Kendall, Y. Gal and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Salt Lake City, Utah, USA, pp. 7482–7491, 2018.