ConViTML: A Convolutional Vision Transformer-based Meta-Learning Framework for Real-Time Edge Network Traffic Classification

Lu Yang, Songtao Guo, *Senior Member IEEE,* Defang Liu, Yue Zeng, Xianlong Jiao, Yuhao Zhou

***Abstract*—Traditional traffic classification methods struggle to identify emerging network traffic due to the need for model retraining, which hampers the real-time response of deployed edge devices. Furthermore, emerging network traffic samples are often scarce, traditional methods often treat a session as a single image, thereby overlooking essential structural features. These factors can result in poor generalization ability of the trained model. To overcome these challenges, we propose ConViTML (Convolutional Vision Transformer-based Meta-Learning), a real-time end-to-end network traffic classification framework that employs meta-learning to avoid model retraining. We propose a novel feature extraction network, Convolutional Visual Transformer (ConViT), merging Convolutional Neural Network (CNN) and Visual Transformer (ViT). ConViT can directly extract low-dimensional discriminative features containing basic and structural features of the session, which is vital for improving detection accuracy and accelerating convergence in a data-scarce environment. Furthermore, we employ a Packet-based Relation Network (PRN) to analyze the matching degree of support samples and query samples. Therefore, accurate classification in novel traffic identification tasks can be achieved with just a few labeled samples, eliminating extensive data collection and labeling operations. Finally, we replace various feature extractors and compare our approach with the classic meta-learning framework Relation Network (RelationNet). Extensive experimental results demonstrate that ConViTML outperforms others with various performance indicators.**

***Index Terms*—Meta-learning, network traffic classification, edge computing, visual transformer.**

I. Introduction

With the continuous emergence of new industrial applications, the real-time traffic of the backbone network of the Industrial Internet of Things (IIOT) shows an explosive growth trend. It not only brings great difficulties to network service quality assurance but also to network security management [1]–[3]. Traditional cloud data center networks are difficult to meet the real-time, security and reliability requirements of IIoT for massive data transmission and processing. Edge intelligence is a promising technology in which endpoint devices can send only the information needed for cloud computing instead of raw data. It helps reduce the cost of cloud infrastructure connectivity and data transmission [4]–[6]. Then again, monitoring and managing network traffic can provide fast channels for priorities by categorizing applications, which improves the quality of network service. Moreover, it can identify malicious traffic and realize network security management, which is of great significance for network operation and maintenance management. Thus deploying network traffic classification services on edge nodes can quickly make decisions and trigger appropriate management actions.

In general, network traffic classification methods mainly include port-based [7], payload-based, Machine Learning-based (ML-based), and Deep Learning-based (DL-based) methods [8]. However, due to the widespread adoption of dynamic port technology and the increasing demands for network performance, the progress of port number-based and DPI methods has been impeded [9]. Classical ML-based methods are based on that the traffic characteristics generated by different types of applications is variable, such as Inter-Arrival-Time (IAT), flow duration, and other network flow features. However, these methods often rely on manual feature design to determine which feature set is most effective for the classification of network traffic in different sub-domains. This limitation restricts their generality. Methods combined with Deep Learning (DL) have become mainstream [10], [11]. Unlike traditional ML-based classification methods which are based on manual feature extraction, DL-based classification methods utilize a large number of labeled traffic samples (preprocessed TCP/UDP bidirectional flows) and automatically extract relevant features [12]–[15]. This enables end-to-end traffic classification, where raw network traffic serves as input and the output is the corresponding category label. This approach partially addresses the problem of manual feature design.

However, there are still several limitations and challenges for the practical deployment of DL models. Firstly, due to the low frequency of malicious attacks, obtaining malicious network traffic samples is more challenging compared to normal traffic samples. This leads to an imbalanced data distribution for different types of network traffic [16]. Furthermore, the emergence of unknown network threats, such as Advanced Persistent Threats (APTs) and zero-day vulnerabilities, appears to be never-ending [17]. These emerging threats lead to constantly evolving challenges in traffic classification and detection. Traditional DL methods heavily rely on massive amounts of data. Insufficient training samples can result in serious overfitting. Moreover, when using a trained model to identify novel traffic, the accuracy may drop significantly [18]. For edge devices where trained models are deployed, han-

L. Yang, S. Guo, X. Jiao are with the Key Laboratory of Dependable Service Computing in Cyber-Physical-Society (Ministry of Education), and college of computer science, Chongqing University, Chongqing, China, 400044.

D. Liu is with Key Laboratory of Biorheological Science and Technology, Ministry of Education, College of Bioengineering, Chongqing University, No.174, Shapingba Main Street, Chongqing 400044, China

Y. Zeng is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

Y. Zhou is with the College of Electronic Information and Engineering, Southwest University, Chongqing 400715, China

The corresponding author: Songtao Guo, Email: guosongtao@cqu.edu.cn; Defang Liu, Email: liudf@cqu.edu.cn

dling this situation becomes a significant challenge. The need for model updates requires substantial computing resources and a large amount of storage to accommodate large-scale labeled training data, while edge devices typically have limited resources such as CPU and memory [19]. Additionally, the process of collecting and labeling novel traffic data is time-consuming and labor-intensive. As a result, edge devices are unable to quickly identify novel attacks, leading to significant security risks.

Meta-learning has been viewed as a promising solution for enabling models to learn latent patterns in data with just a few training examples. The key to meta-learning is to learn an embedding network capable of transforming raw inputs into appropriate representations. These representations can effectively capture the underlying correlations in both the support and query sets, empowering the model to swiftly generalize to new tasks [20]. Introducing meta-learning brings several notable benefits. Firstly, it achieves satisfactory performance on new tasks with a minimal number of labeled examples, avoiding the labor-intensive process of data collection, cleaning, and labeling. Additionally, there is no requirement to retrain the model, and edge devices can promptly identify novel attacks, ensuring system security. These advantages collectively contribute to the practical deployment of network traffic identification systems in edge environments.

Meta-learning has garnered significant attention in various image recognition domains, but its application in network traffic classification is still limited. In the context of meta-learning with limited training samples, efficiently extracting discriminative information through task-based embeddings becomes crucial. However, most existing studies tend to directly convert raw traffic data into images and use Convolutional Neural Networks (CNN) to extract spatial features [18], [21], [22]. Furthermore, traditional global pooling operations in CNN may lead to the loss of valuable local information. Network traffic flows are typically composed of interconnected data packets arranged in chronological order, and the aforementioned methods fail to capture the temporal relationship between packets. Some researchers have attempted to combine CNN and Long Short-Term Memory (LSTM) to jointly extract spatio-temporal features [23], [24]. Nevertheless, this approach demands considerable computing resources and memory.

To address the aforementioned challenges, we propose a lightweight real-time end-to-end network traffic classification framework, ConViTML (Convolutional Vision Transformer-based Meta-Learnin), which is based on Visual Transformer (ViT) and CNN. This framework facilitates the rapid identification of novel traffic at the network edge without the need for retraining. Our method involves several key steps. Firstly, we preprocess the raw traffic data by converting each packet into a fixed-size grayscale image. The resulting sequence of grayscale images corresponds to the sequence of data packets. Then, we utilize a single-layer CNN to extract shallow features from packets, which are treated as packet-level basic features. To prevent the loss of valuable local information caused by multiple global pooling operations, we incorporate ViT, which effectively captures task-related features while suppressing irrelevant ones. To capture the relationship between packets, we treat each packet as a patch, and apply ViT to extract the sequence structure features of traffic while preserving the basic features of the individual packets. Additionally, we design a Packet-based Relation Network (PRN) to measure the similarity between query samples and support samples. This module calculates the similarity between each corresponding data packet, considering packets as individual units, and computes the average relation score as the final metric for classification.

In summary, the contributions of our work can be summarized as follows:

- We propose an end-to-end traffic classification framework for edge environments called ConViTML. In the ever-changing edge network environment, this method can realize the classification of out-of-distribution traffic samples with a small number of support samples without retraining the original model.
- We propose a novel feature extraction model called ConViT, which helps reduce redundant information and enables fast inference. The model can extract low-dimensional basic features and structural features in parallel, and the learned mixed features are more discriminative.
- We propose the PRN that conducts similarity matching between query samples and support samples at the granularity of packets to achieve higher accuracy.
- We evaluate the performance of ConViTML on multiple datasets, and the experimental results show that the framework takes the least training time and achieves the best performance compared to the baselines. Besides, it is very lightweight with a model size of only 2.45MB.

The rest of this paper is organized as follows. In Section II, we review the related works on network traffic classification. In Section III, we formulate the problem. Then we detail the preprocessing steps for the raw traffic data in Section IV. In Section V, we introduce the ConViTML framework, and in Section VI we demonstrate the simulation and evaluation results. In Section VII, we conclude this paper.

II. Related works

There is already quite a lot of literature in this area. According to different working principles, network traffic classification methods can be divided into four methods: port-based, DPI, ML-based, and DL-based [10].

In the early stages, port-based methods [25] and DPI [9] are commonly used. Port-based methods use the correspondence between port numbers and application-layer protocols to classify traffic. For example, HTTP protocol uses port 80 and SSL uses port 443. However, this method is limited by dynamic ports [26]. DPI determines network traffic classes by analyzing whether the payload of packets matches the signature library. This method requires pre-establishing an application layer feature identification rule base for network traffic and verifying whether it matches the feature identification rules in the base by analyzing the key control information in the payload. However, the complete network payload analysis not only has a high computational cost but also may involve user privacy disputes and data security leakage issues [27]. To overcome these limitations, researchers began to seek other traffic classification methods with lower overhead and better performance.

So far, some literature employed machine learning to design models that effectively classify network traffic for security. For example, [28] used a combination of Support Vector Machine(SVM), Decision Tree (DT), and Naive Bayes classifiers to reduce the false alarm rate. [29] used Particle Swarm Optimization(PSO) to further divide the recognition space to improve the classification accuracy and optimization speed. [30] proposed an Intrusion Detection System(IDS) based on binary PSO and Random Forest (RF), where PSO is used to find the best appropriate features and RF is used as a classifier. [31] proposed an IDS based on K-Nearest Neighbors (KNN), which employs a percentage-based simplified neighborhood technique instead of group clustering. However, network traffic classification based on machine learning often requires a manual selection of traffic features, which requires expert experience. At the same time, the rationality of feature extraction seriously affects the accuracy of classification.

Representation learning, also known as feature learning, is a technique of DL that automatically extracts features from different classes of network traffic. In 2015, Wang et al. initially proposed the similarity between images and TCP flow payloads and imagined that a payload record is a picture or a document, and each byte is a pixel or a word [32]. Based on this work, numerous studies have emerged that involve converting network traffic into grayscale images and subsequently employing them in conjunction with various DL models. This also motivates us to adopt the method of converting network traffic into images. [2] proposed a lightweight DDoS attack detection model based on CNN, Lucid. [33], [34] proposed an IDS based on CNN, which employed PCA and Auto-Encoder(AE) respectively to remove redundant features to improve the model's performance and convergence speed. [35] focused on the problem of classifying encrypted traffic. It converted encrypted network traffic data into intuitive images and employed CNN to recognize traffic categories. Additionally, some literature treats network traffic as other forms of data as input. [36], [37] treated network traffic as the time-series data and extracted features using Stacked Sparse Autoencoders (SSAE) and the enhanced Multimodal DL-based Mobile Traffic Classification (MIMETIC) framework respectively. [38] mapped network traffic to graph representations as input for the Graph Neural Network (GNN) architecture.

While these models can be deployed on edge nodes, identifying out-of-distribution network traffic samples requires collecting, cleaning, and labeling samples, and retraining the models based on these samples. This process is often time-consuming, labor-intensive, and requires a significant amount of computational resources. Moreover, out-of-distribution traffic samples are typically limited in number, making accurate feature representation particularly crucial. In edge environments, there are often challenges such as limited resources and sensitivity to latency. Therefore, we designed an end-to-end traffic classification framework that parallelly extracts low-dimensional basic features and structural features to obtain discriminative hybrid features. This design effectively removes redundant information, enhances classification accuracy, and reduces training time. Moreover, meta-learning is adopted to alleviate the contradiction between the massive overhead caused by model updating when dealing with out-of-distribution problems and the limited resources of edge devices.

III. Problem Formulation

A. Motivation for introducing meta-learning

We focus on metric-based meta-learning, where the principle involves learning a metric or distance function to measure the similarity between different samples for the purpose of classifying data samples based on their similarities. Generally, it includes a feature extraction network and a metric function. Firstly, samples are mapped into a smaller-dimensional embedding space using the feature extraction network. Then, a metric function is employed to calculate the similarity between the support set data and the query set data within the embedding space, and a certain classification strategy is applied to categorize the query set data. In this paper, we utilize the PRN as the metric function to measure the relation scores between samples in the query set and the support set. Higher relation scores indicate a higher likelihood that the query set samples and support set samples belong to the same category. By learning how to measure similarity between samples, meta-learning requires only a few support sets, thereby reducing the reliance on a large amount of labeled data [39].

It is noted that in network traffic classification, the constant emergence of new attack types, including zero-day attacks, poses a significant challenge. In fact, traditional DL-based methods are unable to detect new attacks. This is because traditional DL-based methods are trained using a large amount of labeled data to learn the mapping relationship between input data and corresponding labels. The model weights are fixed after training, for instance, each dimension of the output layer corresponds to the probability of each class in the training dataset. When the trained model is used to detect novel traffic samples such as zero-day samples, it will become ineffective because the training dataset lacks samples of this type.

In contrast, meta-learning operates on a task-level training unit, where each task consists of a support set $\mathcal{D}_S$ and a query set $\mathcal{D}_Q$. Unlike traditional DL methods, meta-learning aims to enable models to quickly learn new tasks based on existing knowledge, thus avoiding the computational cost associated with retraining the model for each task switch [40]. The main idea is to train the model to learn a similarity function that measures the similarity between support set and query set samples. When the trained model is required to identify novel traffic samples such as zero-day attacks, it only needs to add a few labeled samples to the support set. By comparing the similarity, the model can determine whether the current traffic sample is a zero-day attack.

TABLE I: Related Works in Network Traffic Classification Tasks

Related Work	Port-based	DPI	ML-based	DL-based	Out of Distribution
[25]	✓	✗	✗	✗	✗
[9]	✗	✓	✗	✗	✗
[28]–[31]	✗	✗	✓	✗	✗
[32]–[38]	✗	✗	✗	✓	✗
This paper	✗	✗	✗	✓	✓

B. Task construction process

We construct the basic unit of meta-learning according to the following steps. First, we use $\mathcal{D}$ to represent the multidimensional array dataset generated by the preprocessed raw traffic data, and (x_t, y_t) to represent a data sample of $\mathcal{D}$. $x_t \in \mathbb{R}^{m\times d_1\times d_1}$ is the input of the model, and y_t is the corresponding class label. In order to achieve out-of-distribution sample classification, it is necessary to learn the extrapolation rules. We therefore divide the original dataset into two mutually exclusive subsets, the meta-training set $\mathcal{D}^{train}$ and the meta-testing set $\mathcal{D}^{test}$. They are all sets consisting of sampling tasks. We use the $\mathcal{D}^{train}$ set to train the model, while the tasks in $\mathcal{D}^{test}$ are used to measure the final performance of the model.

In addition, the construction process of task $\mathcal{T} = \{\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{Q}}\}$ is as follows, we select M traffic types from $\mathcal{D}^{train}$ by random, and each type has K labeled samples. These $M\times K$ samples form a support set $\mathcal{D}_{\mathcal{S}} = \{(x_1,y_1),(x_2,y_2),\dots,(x_{M\times K},y_{M\times K})\}$. Additionally, select T samples from these M types to form a query set $\mathcal{D}_{\mathcal{Q}}$, which can be denoted by $\mathcal{D}_{\mathcal{Q}} = \{(x_{M\times K+1},y_{M\times K+1}),\dots,(x_{M\times K+T},y_{M\times K+T})\}$. In general, problems of this form are called M-way K-shot classification problems.

IV. Network Traffic Data Preprocessing

The public datasets of network traffic can be divided into raw traffic datasets and processed flow features datasets. In actual use, network traffic data is mostly saved as a file in pcap or pcapng format. To upgrade the practicality, we design an end-to-end classification framework that uses the raw traffic dataset as the input to the model. The size of the pcap file is variable, but neural network models require input data of a uniform format and size. Wang et al. proposed that each byte can be regarded as a pixel or a word, and the flow payload record can be regarded as the fixed-size image [32]. Numerous studies utilizing various DL models have demonstrated the effectiveness of this approach [2], [33], [34]. Therefore, to better input the network traffic into the DL model for training, our proposed framework transforms the raw network traffic data into a grayscale image through a series of preprocessing operations, which can considerably speed up the later analysis and mining process.

From the granularity of the target for network traffic classification, it can be further divided into packet-level, flow-level, and session-level from small to large. The smaller the granularity, the more attributes that characterize the classification object, and the greater the preprocessing workload required. Flow level classification focuses on groups of packets with the same quintuple (source IP, source port, destination IP, destination port, protocol). In this paper, we use the most widely used session (bidirectional flow) in current research as the input object of the neural network.

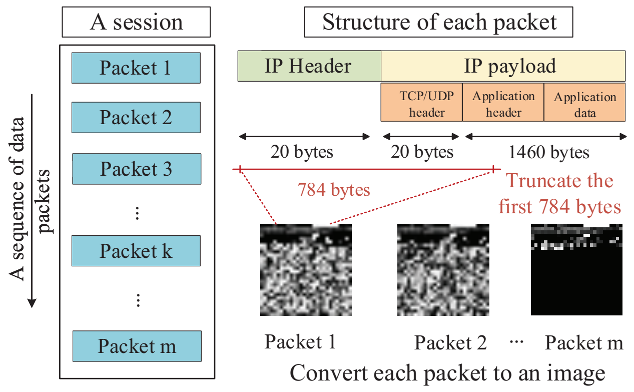


Fig. 1: Network traffic preprocessing

A session is composed of several data packets and packet headers. The split file contains too much redundant information, which not only affects the final classification accuracy but also increases the computational complexity of the model. Data packets with indicators such as SYN, ACK, FIN are a case in point. During the three-way handshake, these packets are only used to establish a TCP connection and do not carry the payload. On the other hand, IP addresses and MAC addresses are used as logical addresses and physical addresses of network devices, have no bearing on the classification of traffic data. Therefore, we should anonymize network traffic. Besides that, since TCP and UDP headers have different lengths (20 bytes for TCP vs. 8 bytes for UDP), we pad the end of the UDP header with zeros to ensure the same length. In summary, the data preprocessing procedure includes the following four steps, as shown in Fig. 1:

(1) **Traffic split**: The pcap file is split into smaller files based on the criterion of session.

(2) **Traffic clear**: Remove duplicated packets and packets that do not carry payloads. Wipe off IP and MAC addresses from headers.

(3) **Uniform data length**: Pad the UDP header to 20 bytes, and only the first m data packets are reserved. The influence of the value of m on the classification results will be studied

in Section VI.

(4) **Convert the packets to images**: Split each session into data packets and further convert each packet into a grayscale image.

(5) **Convert the packets to images**: According to SectionIII, the original network traffic dataset is constructed as a task-based training dataset for meta-learning.

Acronyms	Explanation
PRN	Packet-based Relation Network
RelationNet	Relation Network
DPI	Deep Packet Inspection
ViT	Visual Transformer
APTs	Advanced Persistent Threats
ConViTML	Convolutional Vision Transformer-based Meta-Learning
SVM	Support Vector Machine
DT	Decision Tree
IDS	Intrusion Detection System
PSO	Particle Swarm Optimization
RF	Random Forest
KNN	K-Nearest Neighbors
AE	Auto-Encoder
SSAE	Stacked Sparse Auto-Encoder
RNN	Recurrent Neural Network
SAE	Sparse Auto-Encoder
MHA	Multi-Head Attention
FF	Feed Forward
GELU	Gaussian Error Linear Unit
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
ACC	Accuracy
DR	Detection Rate
FAR	False Alarm Rate
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
GNN	Graph Neural Network
MIMETIC	Multimodal DL-based Mobile Traffic Classification

TABLE II: Explanation of Acronyms

V. CONVITML FOR NETWORK TRAFFIC CLASSIFICATION

When faced with network traffic from previously unseen classes and limited labeled samples, the scarcity of data can hinder the generalization ability of traditional DL models. In the past, CNN has been commonly used for feature extraction in traffic classification [11], [41]. However, CNN is limited in processing local areas of a sample, and due to the fixed kernel size in convolution operations, it may not efficiently capture long-range dependencies, resulting in reduced accuracy during learning from limited samples. Additionally, multiple pooling operations often result in the loss of essential local information.

ViT introduces a novel concept of partitioning an image into multiple equally-sized patches, followed by the application of the transformer's self-attention mechanism on each patch to emphasize task-related features and capture dependencies between patches [42]. However, ViT's patchify stem can be viewed as a non-overlapping stride convolution implementation, which restricts its ability to effectively extract local spatial information. Additionally, the complex attention mechanism and model design of ViT make it less efficient compared to CNN.

Traffic consists of a sequence of data packets, where the interaction between the sender and the receiver is represented by adjacent data packets. We aim to capture the temporal relationship between data packets and harness the strengths of both the local modeling capability provided by CNN and the global modeling capability offered by ViT [43]. Through the combination of these two components, we can extract deeper and more discriminative features, enhancing the effectiveness of our traffic classification approach.

The proposed framework ConViTML, which consists of three key components as follows.

A. Package-Level Patchify Stem

The preprocessed input of ConViTML, denoted as $x_t \in \mathbb{R}^{m\times 1\times d_1\times d_1}$, is fed into the network. Each packet p_i is treated as a patch, and we extract shallow low-dimensional features using a convolutional layer consisting of the sequential module with two operations. Firstly, we use a 2D convolutional layer with a single input channel and a single output channel. The convolutional kernel size is set to 3, with a stride of 1 and no padding (padding is set to 0). Following this, we perform a 2D max pooling operation with a pooling window size of 2, sliding over the input with a stride of 2. This simple convolutional layer is utilized to extract packet-level basic features and reduce dimensionality, thereby effectively reducing the number of parameters involved. The packet-level basic feature vector $w_t \in \mathbb{R}^{m\times 1\times d_2\times d_2}$ of each session extracted from the CNN can be expressed as

$$w_t = f\left(x_t, \theta\right), \tag{1}$$

where θ denotes the training parameters of the CNN and x_t represents an input sample. Since the transformer takes sequence data as input, we flatten each data packet after dimensionality reduction to sequence format of size d_2^2. Subsequently, we utilize ViT to model the sequence information of packets within the session. To facilitate this process, learnable position embeddings are incorporated to capture the positional relationships among the packets. Furthermore, to streamline the subsequent processing in the PRN, we remove the class token. This process can be expressed as

$$z_t = W \times w_t + PE, \tag{2}$$

where W is the linear flattening operation and $PE \in \mathbb{R}^{m\times d_2^2}$is the positional embedding, $z_t \in \mathbb{R}^{m\times d_2^2}$ is the final session embedding after the package-level pathify stem.

B. Transformer Encoder Block

Currently, session embedding z_t encompasses the basic features of packets. To further extract the structural information

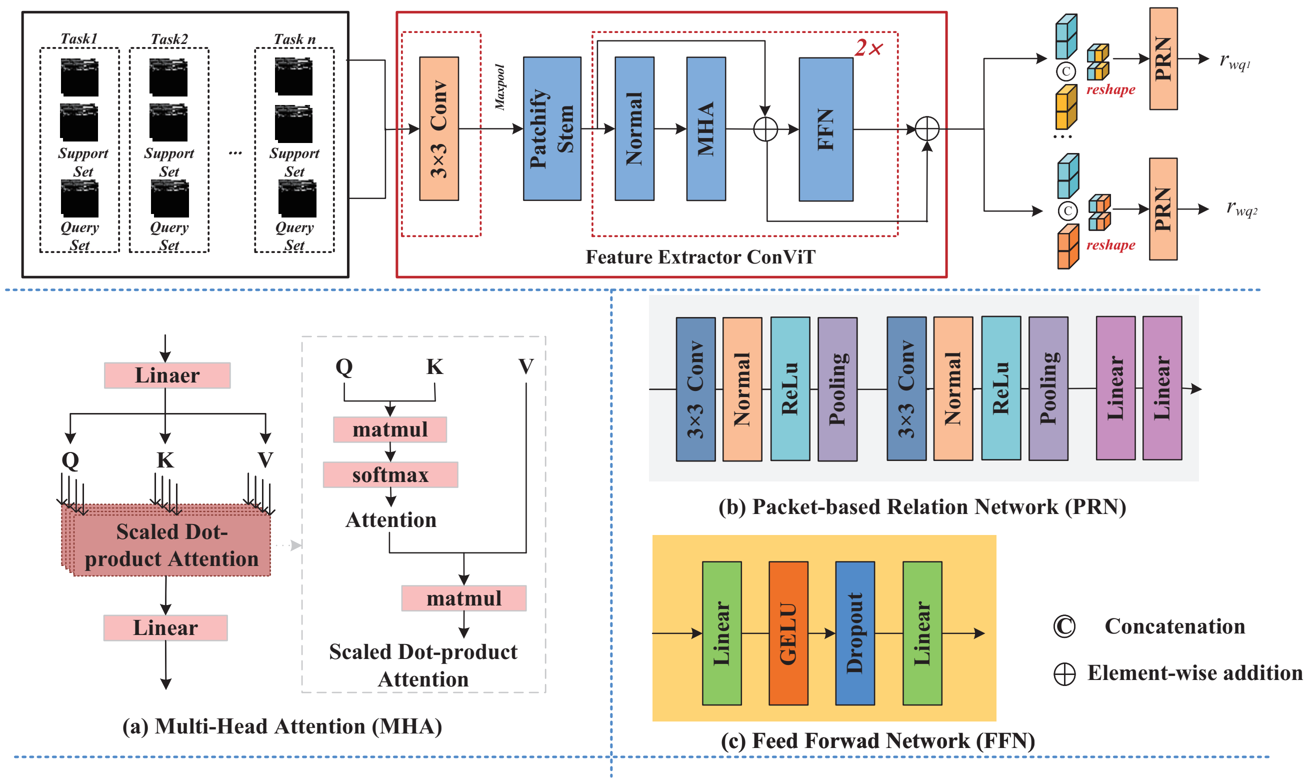


Fig. 2: The overall architecture of ConViTML.

of the session, we adopt the classic transformer encoder block of the ViT network, which comprises a Multi-Head Attention (MHA) block and a Feed Forward (FF) block. In our implementation, we apply the normalization layer solely before the MHA block while not utilizing the masking mechanism. The MHA module's architecture is depicted in the accompanying Fig.2(a). Assuming a total of k heads, we obtain k sets of query (Q), key (K), and value (V) matrices through this operation. This process can be mathematically represented as

$$Q = z_t W_Q, K = z_t W_K, V = z_t W_V. \tag{3}$$

Here, W_Q, W_K, and W_V represent shared learnable projection matrices used across all data packets. The correlation between the query (Q) and the key (K) is computed through a dot product operation, followed by the application of a softmax activation function. This generates the attention matrix, which is then used as the weight for the corresponding value (V) in the attention mechanism. This process can be expressed as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{4}$$

where d is the dimensionality of the key K and query Q, represents a scaling factor used to avoid excessively large values resulting from the dot product operation. This scaling helps prevent the gradients from becoming too small after passing through the softmax function. Then, we concatenate the outputs of each attention head and pass them through a linear layer to obtain the final output of the MHA block. This process can be expressed as

$$\begin{aligned} \text{head}_j &= \text{Attention}\left(z_t W_j^Q, z_t W_j^K, z_t W_j^V\right), j \in (0, k] \\ \text{MHA}(Q, K, V) &= \text{ Concat }\left(\text{head}_1, \ldots, \text{head}_n\right) W^O, \end{aligned} \tag{5}$$

where W_j^Q, W_j^K and W_j^V represent the mapping matrices of the j-th attention head, and W^O denotes the parameter matrix of the final linear layer.

The FF block is composed of two linear layers, incorporating a non-linear transformation through an activation function. Firstly, the input features are passed through the first linear layer to map them into a higher-dimensional space. Subsequently, the mapped features undergo a non-linear transformation using an activation function, often the Gaussian Error Linear Unit (GELU). Finally, the transformed features are mapped back to the original dimensions using the second linear layer. The specific structure of the FF block is depicted in Fig.2(c).

The FF block is designed to flexibly and adaptively perform non-linear transformations on the features at each packet, leveraging the Multi-Layer Perceptron (MLP) structure. This allows for the capture of richer feature representations and enhances the expressiveness of the model.

To preserve and propagate important information, residual connections are employed within the FF block. These connections enable the direct addition of the original input to the output of the module, facilitating efficient information transfer and propagation throughout the network.

C. Packet-based Relation Network

The Relation Network (RelationNet) [39] is employed to address the few-shot classification problem. Its core is to calculate the distance between two samples by constructing a neural network and analyzing their matching degree [39]. In this work, we propose the PRN, with the aim of achieving more precise classification with finer granularity. For each packet in a query session, PRN calculates its similarity score with the aligned packet of the support session. It then proceeds to average the similarities of all packets in the query session, leading to the derivation of the sample-to-class similarity. Specifically, We represent the final representation for each session as $s_t \in \mathbb{R}^{m\times d_2^2}$, where m is the number of packets in the session, and d_2 is the embedding dimensionality. Additionally, we denote the final representation for the support set samples as $\{(s_1, y_1), (s_2, y_2), \ldots, (s_{M\times K}, y_{M\times K})\}$, and for the query set samples as matrix $\{(s_{M\times K+1}, y_{M\times K+1}), \ldots, (s_{M\times K+T}, y_{M\times K+T})\}$.

Next, we add the feature vectors obtained from the K samples of the same class in the support set to obtain the feature vector of the corresponding class. The set of feature vectors for all classes is denoted as $\mathcal{D}'_{\mathcal{S}} = \{(s'_1, y'_1), \ldots, (s'_w, y'_w), \ldots, (s'_M, y'_M)\}$. Afterwards, we reshape all samples into $m \times 1 \times d_2 \times d_2$ in order to facilitate the input of the PRN. Then, we concatenate the two latent vectors as $\mathcal{C}(s'_w, s_q) \in \mathbb{R}^{m\times 2\times d_2\times d_2}$, with $w \in [1, M], q \in [M \times K + 1, M \times K + T]$.

The combined feature map of s'_w and s_q is input into the relation module g_ϕ on a per-packet basis to calculate the distance between a single query input and each support sample. This process can be expressed as:

$$r_{w,q} = \text{Average}\left(\sum_{i=1}^{m} g_\phi\left(\mathcal{C}\left(p_i^{w,\prime}, p_i^q\right)\right)\right), \tag{6}$$

where $r_{w,q}$ is the relation score between the sample feature vector s_q and the class feature vector s'_w. This score is computed as the average of all packet-level relation scores. $p_i^{\mathrm{w},\prime}$ represents the ith packet embedding in s_q, and p_i^q denotes the ith packet embedding in s'_w. $\mathcal{C}(\cdot) \in \mathbb{R}^{2\times d_2\times d_2}$ represents the embedding after concatenating the corresponding packets. A higher relationship score indicates a greater similarity between s'_w and s_q.

When s'_w and s_q belong to the same class, that is $y'_w = y_q$, the value of relation score $r_{w,q}$ is closer to 1; however, when s'_w and s_q belong to different class, that is $y'_w \neq y_q$, the value of relation score $r_{w,q}$ is closer to 0. In fact, what is judged by the relation score is the possibility that s'_w and s_q belong to the same class. Therefore, the predicted label of the sample can be obtained by

$$\hat{y}_q = \text{argmax}\left(r_{w,q}\right), q \in [M \times K + 1, M \times K + T]. \tag{7}$$

D. ConViTML for Network Traffic Classification

We use the Mean Square Error (MSE) to supervise the similarity score, and the loss function can be expressed as

$$\mathcal{L} = \sum_{w=1}^{M} \sum_{q=M\times K+1}^{M\times K+T} \left(r_{w,q} - \mathbf{1}\left(y_w == y_q\right)\right)^2. \tag{8}$$

The classification problem generally uses cross-entropy, but since the final score is a 0 to 1 relationship score, it can also be regarded as a regression problem, so the MSE is used as the loss function.

The input of ConViT network is multiple tasks. Algorithm 1 gives the loss calculation process when inputting a task. First, the CNN extracts low-dimensional shallow basic features w_t, which are then input into ViT to obtain the final session embedding s_t. This process preserves the structural features of session while retaining the package-level basic features. Furthermore, we concatenate the embeddings of each sample in the query set and the embeddings of each class at the packet level as inputs for the PRN, in order to obtain the final predicted label and loss.

Algorithm 1 The training process of one task on ConViTML

```
Require: A task for training $\mathcal{T} = \{\mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{Q}}\}$. Batch size $T'$.
Ensure: The loss $\mathcal{L}$ for back propagation.
1: $\mathcal{L} \leftarrow 0$
2: for i = 1 to $T'$ do
3:     for $x_s$ in $\mathcal{D}_{\mathcal{S}}$ do
4:         Transfer $x_s$ into embedding vector $s_s$ via ConViT.
5:     end for;
6:     Add the feature vectors of the same class in the support
       set to get the final feature vector of each class $\mathcal{D}'_{\mathcal{S}} = \{(s'_1, y'_1), \ldots, (s'_w, y'_w), \ldots, (s'_M, y'_M)\}$.
7:     for $x_q$ in $\mathcal{D}_{\mathcal{Q}}$ do
8:         Transfer $x_q$ into embedding vector $s_q$ via ConViT.
9:         for $s'_w$ in $\mathcal{D}'_{\mathcal{S}}$ do
10:            Cascade $s'_w$ and $s_q$ into a new vector and
       calculate the similarity between $s'_w$ and $s_q$ by Eq.6.
11:            Predict the class label by Eq.7.
12:        end for;
13:    end for;
14:    $\mathcal{L} \leftarrow \mathcal{L} + \text{MSE}(r_{w,q}, y_q == y_w)$.
15: end for
```

The overall ConViTML network architecture is shown in Fig. 2. The network consists of a ConViT feature extraction model and a PRN. To begin with, a one-layer CNN serves as the basic feature extraction layer. Specifically, this convolutional block includes a 3×3 convolutional kernel with a single filter and a max pooling layer with the size of 2×2. Incorporating ideas from ViT, we utilize the package-level patchify stem to introduce location information for each packet in a session while removing the class token for subsequent input. The ViT component has a depth of 2, consisting of both the MHA and FF modules. The PRN comprises two convolutional blocks and two linear layers. Each convolutional block comprises a 2D convolutional layer with a kernel size of 3, followed by a regularization layer, a ReLU activation function, and a maximum pooling layer. The network concludes with a sigmoid function, which generates the final relation score as the output.

E. Time Complexity Analysis

Due to the package-level patchify stem only involving a Conv2D layer, the time complexity for this segment is $O\left(m \times d_2^2 \times k_1^2\right)$, where k_1 denotes the kernel size. The depth of the ViT is 2, resulting in a time complexity of $O\left(2 \times m \times 4d_2^2 \times 2(d_2^2)^2\right)$. The PRN comprises three Conv2D layers, leading to a time complexity of $O\left(m \times d_3^2 \times k_2^2 \times C_{in} \times C_{out}\right)$, where d_3 indicates the final output dimension of the PRN. C_{in} represents the number of input channels, and C_{out} represents the number of output channels.

VI. Performance Evaluation

In this section, we first introduce the dataset used, experimental configuration and evaluation metrics used, and then use the raw network traffic data to evaluate GAEML.

A. Dataset

In evaluating the efficacy of the proposed end-to-end network traffic classification framework, certain aspects warrant attention. Firstly, concerning dataset selection, it is crucial to utilize raw network traffic datasets as opposed to processed or extracted files. Secondly, for precise evaluation result values, it is essential that each raw traffic sample be associated with an appropriate label. However, the KDD'99, UNIBS, WIDE, CIDS2017 and other datasets commonly used in this area are mostly processed traffic feature datasets and the raw network traffic data contained in them do not have corresponding labels. If we label the raw traffic data according to the attack time period specified in the dataset specification, noise may be introduced, resulting in poor training results. Therefore, the USTC-TFC2016 dataset [21] and the CIC IoT 2022 dataset [44] are used for evaluation in this paper. The USTC-TFC2016 dataset consists of two parts, one is 10 kinds of malicious traffic collected from the real environment by researchers from CTU University, and the other is 10 kinds of normal traffic collected by IXIA BPS equipment. We counted the number of sessions in the dataset, and the results and their corresponding network traffic names are shown in Table III.

CIC IoT 2022 is a state-of-the-art dataset collected by the Canadian Institute for Cybersecurity. It includes data for profiling, behavioral analysis, and vulnerability testing of various IoT devices with different protocols, such as IEEE 802.11, Zigbee-based, and Z-Wave. The dataset is collected on 60 IoT devices deployed at the edge of the network. The main target of the CIC IoT 2022 dataset project consists of conducting and capturing the network terrific of devices undercurrent and important attacks in the IoT environment, which consists of two different attacks, Flood and RTSP-Brute Force. We extract 2000 sessions in these two different attacks and 4000 sessions of normal traffic to reconstruct a new dataset and train a model on it to evaluate the performance of the proposed framework on binary classification problems.

We first need to preprocess the pcap files. First, according to the content of Section IV, we perform operations such as traffic split, traffic clear, uniform data length, and convert the packets to images. Since we obtained the average number of bytes per packet according to our statistics is 673 bytes, in order to obtain the packet header and part of the payload, we intercept the first 784 bytes of each packet, that is, $N = 784$. If the length of the data packet is less than 784, it is padded with 0. We list the grayscale images of packets that appear more frequently in each type of traffic, as shown in Fig. 3. Here only one grayscale image is listed in Factime because there is only one packet in each flow. It can be found that although the grayscale images of different types are quite different, there are still some grayscale images that are very similar. Therefore, extracting the sequence relationship between the packets can effectively improve the distinguishability of the extracted features.

B. Metric

Since the basic unit of meta-learning is a task, the metric we choose is to evaluate the completion of tasks. To assess the performance of the proposed framework, we apply four evaluation metrics, Accuracy (ACC), Detection Rate (DR), False Alarm Rate (FAR), and F1 Score. We first get the number of True Positive (TP) instances, True Negative (TN) instances, False Positive (FP) instances, and False Negative (FN) instances on the reserved test set. Based on these values, we calculate the exact values of the four metrics according to the following formulas, which are used to evaluate the overall performance of the proposed framework. For each experiment, we generate multiple tasks and calculate the values of these four metrics. Our objective is to enhance the ACC metric, DR metric, and F1 Score metric while maintaining a low FAR metric.

$$\text{ACC} = \frac{TP+TN}{TP+FP+TN+FN} \tag{9}$$

$$\text{FAR} = \frac{FP}{TN+FP} \tag{10}$$

$$\text{DR} = \frac{TP}{TP+FN} \tag{11}$$

$$\text{F1 Score} = \frac{2 \times TP}{2 \times TP+FN+FP} \tag{12}$$

C. Experimental Settings

The hardware and software platforms used in the experiment are as follows: 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz, PyTorch1.10.0, Ubuntu 20.04 LTS, CUDA 11.4, NVIDIA Geforce RTX 3090 Founders Edition. We follow the split introduced by [39], where the sample categories used for training and testing are orthogonal. For the USTC-2016 dataset, we select 15 traffic categories for training, reserving the remaining 5 categories solely for monitoring generalization performance. Similarly, for the CIC IOT 2022 dataset, we specify 2 categories for testing and use the rest of the categories for training. During network training, we configure

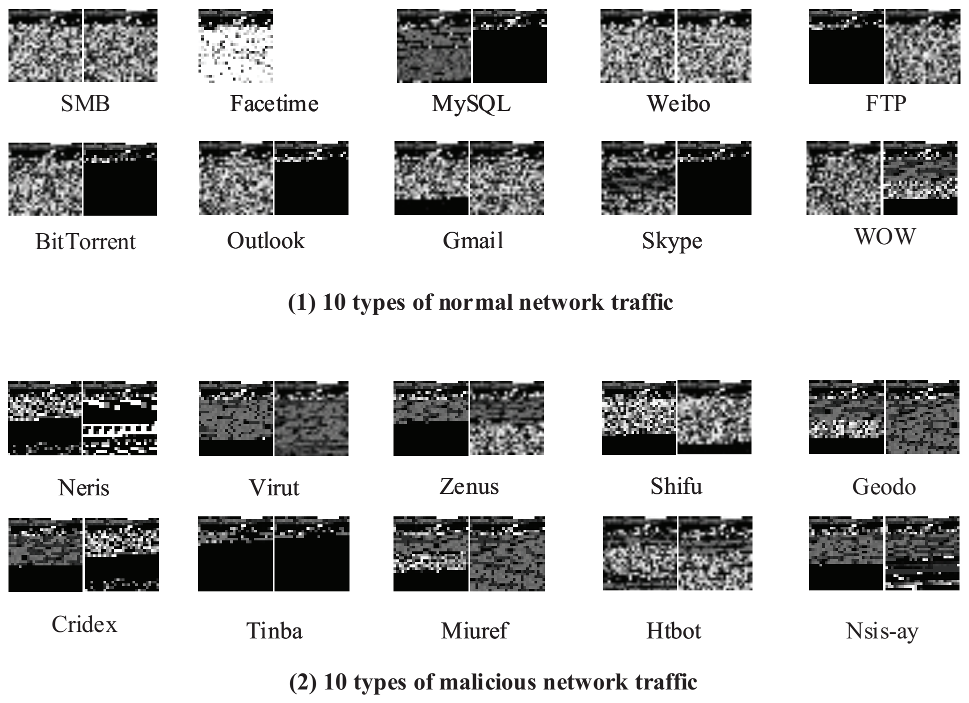


Fig. 3: Grayscale representation of packets

TABLE III: USTC-TFC2016

Name	Type	Samples	Class	Name	Type	CTU num	Samples
BitTorrent	N	7502	P2P	Cridex	M	108-1	8197
Facetime	N	6000	Voice/Video	Geodo	M	119-2	6690
FTP	N	6319	Data Transfer	Htbot	M	110-1	5952
Gmail	N	5111	Email	Miuref	M	127-1	4952
MySQL	N	7026	Database	Neris	M	42,43	8425
Outlook	N	7475	Email	Nsis-ay	M	53	6033
Skype	N	6089	Chat/IM	Shifu	M	142-1	9576
SMB	N	5473	Data Transfer	Tinba	M	150-1	8504
Weibo	N	4569	Social Network	Virut	M	54	6138
WOW	N	7592	Game	Zeus	M	116-2	5664

the learning rate to be 0.0001 and utilize the Adam optimizer. Additionally, we choose MSE as the loss function and employ the episode-based training strategy. In this approach, each task is treated as a training instance, and the model is updated task by task [45]. Each task is formed by randomly selecting M categories from the training set and further selecting K samples from each category, forming an M-way K-shot task. During testing, we generate 300 M-way K-shot tasks from the test set to assess the optimal accuracy and other relevant metrics achieved by the trained model.

Based on the aforementioned settings, we conduct multi-class classification model evaluation on USTC-2016 and binary classification model evaluation on CIC IOT 2022, Experiments include

(1) Determining the best representation of network traffic, the number of packets per session usually varies widely, e.g. Outlook only contains two packets per session while Virut contains hundreds of packets. So we have to perform multiple experiments to determine the optimal number of packets. We set the number of data packets from 2 to 30 to evaluate its impact on various evaluation metrics.

(2) Since the input unit of the model in meta-learning is a task, we need to evaluate the impact of different task forms on the system performance. Following the standard settings adopted by most existing meta-learning works, we perform 5-way 5-shot, 5-way 1-shot, 2-way 1-shot, and 2-way 2-shot classification tasks. The query set for the 5-way 5-shot task and the 2-way 1-shot task contains 19 of query samples. The query sets for the 5-way 1-shot task and the 2-way 2-shot task contain 15 query samples. This means, for example, that for a 2-way 1-shot task, each task contains a total of $19\times2+1\times2=40$ samples.

(3) We conduct the ablation study on both datasets, USTC-2016, and CIC IOT 2022. We exclude the ViT network and use a single-layer CNN to extract basic features from the data packets. The primary objective is to demonstrate the impact of the session's structural features on enhancing the performance of ConViTML.

(4) We replace various networks as feature extractors, including ViT, LSTM, and CNN-LSTM. In addition, we choose the classical RelationNet as the benchmark. To demonstrate the lightweight nature of ConViTML, we further compare its model size with four benchmark models.

(5) Compared to existing traffic classification methods, many excellent works have integrated modular components into current network models. However, they are unable to recognize out-of-distribution traffic samples. Therefore, for the specific scenario of edge environments, we employ training time and testing time as additional evaluation metrics to reflect the low latency cost of the framework. We compare our approach with four existing network traffic classification models: LeNet [21], GoogLeNet [46], HAST-II [23], and DFR [47], along with two classical machine learning methods, K-Nearest Neighbor (KNN) and RandomForest, serving as baselines on the datasets.

D. Experimental Results and Analysis

To begin with, the determination of the optimal input format for the ConViTML framework is of paramount importance. We conduct experiments by observing the changes in various indices as the number of data packets increased from 1 to 13, at intervals of 2. Table IV presents the optimal results of various indicators achieved on the USTC-2016 dataset, while Fig.4(a) and Fig.4(b) showcase the corresponding changes in accuracy and loss for this dataset. Additionally, Table V and Fig.4(c) and Fig.4(d) display the results obtained on the CIC IOT2022 dataset. Fig.4(a) and Fig.4(c) clearly demonstrate that the framework's performance is significantly compromised when the number of data packets is set to 1 for both datasets. This decline in performance can be attributed to the insufficient discriminative information contained within a single data packet. Additionally, the grayscale images converted from the first data packet of various traffic often exhibit strong similarities, further hindering effective discrimination. Additionally, as depicted in Fig.4(b) and Fig.4(d) a notable trend emerges with an increase in the number of data packets: the loss decreases at a faster speed. This can be attributed to the additional discriminative information that is introduced with the inclusion of more data packets. Specifically, for the USTC-2016 dataset, the framework's optimal performance is achieved when the number of data packets is set to 5. Beyond this point, its performance starts to deteriorate. Similarly, for the CIC IOT 2022 dataset, the best performance is observed when the data packet number is 7, followed by a decline. This is because, if the set number of data packets is much larger than the actual number available, we pad the missing packets with 0. Unfortunately, this approach blurs the discriminative characteristics between different types of traffic, adversely affecting the overall performance.

Considering factors such as resource constraints and delay sensitivity in the edge environment, we choose 12 and 14 as the optimal number of packets for the two datasets USTC-TFC2016 and CIC IOT2022, respectively. On this basis, we further conduct experiment (2). We perform 2-way 1-shot, 2-way 2-shot, and 5-way 1-shot 5-way 5-shot classification tasks on the USTC-TFC2016 dataset. Meanwhile, we perform 2-way 1-shot, 2-way 2-shot classification tasks on the CIC IOT 2022 dataset. We randomly shuffle the list of network traffic types each time and divide the list into two parts: training set and test set. The network traffic types of the test set will not participate in model training to verify the effectiveness of the model for out-of-distribution problems. For the 5-way task, we use Macro-F1, which involves calculating precision and recall for each category and then averaging to obtain the Macro-F1 score. Test tasks are performed every 5 epochs on the test set. The experimental results, illustrated in Fig.5, clearly demonstrate that regardless of whether it is a 2-way or 5-way task, increased support samples lead to higher performance and faster convergence. However, it is noteworthy that for the USTC-TFC2016 dataset, the 5-way task results in a decline in the framework's performance. We speculate that this decline may be attributed to the reduction in the number of samples in the 5-way task due to an insufficient number of total categories.

Next, to validate the effectiveness of the structural features, we conduct experiments where we remove the ViT component and solely utilize CNN to extract the packet-level basic features while keeping other components unchanged. The results are depicted in Fig. 6. Notably, by removing ViT, we observe an improvement in convergence speed. However, on the USTC-2016 dataset, the accuracy rate reduces by approximately 10 percentage points, and on the CIC IOT 2022 dataset, the accuracy rate declines by about 5 percentage points. Moreover, other indicators also exhibit a decrease. This outcome clearly indicates that the session structural feature plays a significant role in enhancing performance.

E. Comparison and discussion

We select RelationNet as the baseline for comparison. Additionally, we replace different feature extractors to verify the superiority of ConViT. The details are as follows:

(1) RelationNet: It employs a four-layer CNN as its feature extractor, performing direct feature extraction on a per-session basis.

(2) GCN: We utilize a two-layer Graph Convolutional Network (GCN) to capture the context information of each packet.

(3) ViT [42]: We use a classic ViT network with a depth of 4 to directly input the data packet as a patch, without using CNN for dimensionality reduction.

(4) CNN-LSTM: This model first utilizes a two-layer CNN to extract packet-level features, followed by a two-layer LSTM to capture information between packets.

For RelationNet, we directly reshape a session into a multi-channel image, use CNN to extract session-level features, and then combine the sessions to obtain a similarity score. As for GCN, we treat each data packet as a node and connect adjacent packets using undirected edges. Regarding ViT, we only modify its depth without making any structural alterations. The final results are displayed in Table VI.

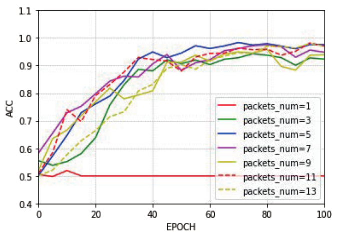
The experimental findings demonstrate that ConViTML achieves the best performance on both datasets, followed

TABLE IV: The impact of the number of packets on performance on USTC-2016

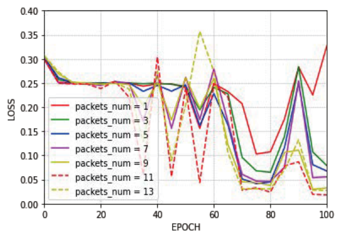
Number of packets	1	3	5	7	9	11	13
ACC(%)	55.50	94.91	99.75	97.66	96.16	98.33	98.32
FAR(%)	46.50	5.83	0.16	2.69	4.73	1.31	1.16
DR(%)	57.50	95.66	99.66	97.33	96.33	98.66	98.62
F1 Score(%)	56.37	94.95	99.74	97.65	96.17	98.83	98.91

TABLE V: The impact of the number of packets on performance on CIC IOT 2022

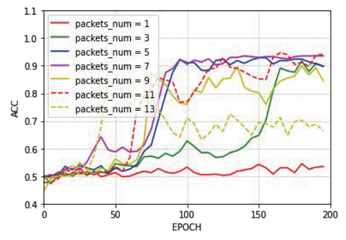
Number of packets	1	3	5	7	9	11	13
ACC(%)	58.25	93.00	93.64	93.54	91.00	95.50	91.50
FAR(%)	39.00	9.00	2.19	1.00	7.57	4.82	12.60
DR(%)	55.50	95.00	96.71	97.04	89.82	89.52	68.05
F1 Score(%)	57.06	93.13	97.76	98.02	83.65	89.37	68.14



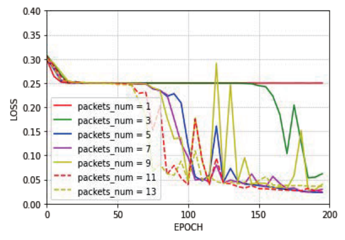
(a) Accuracy on USTC-2016



(b) Loss on USTC-2016

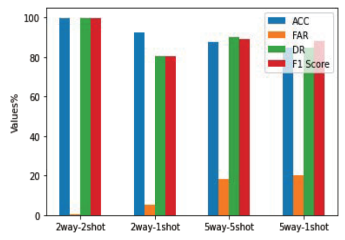


(c) Accuracy on CIC IOT 2022

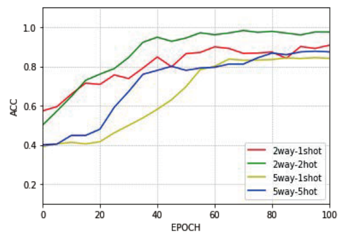


(d) Loss on CIC IOT 2022

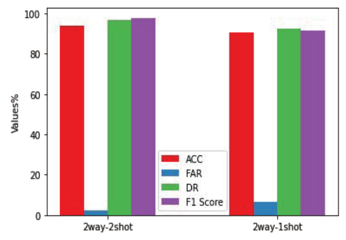
Fig. 4: Accuracy and Loss vs Epochs for different numbers of packets on USTC-2016 and CIC IOT 2022 datasets.



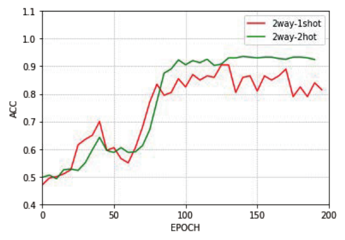
(a) Metrics on USTC-2016



(b) Accuracy on USTC-2016



(c) Metrics on CIC IOT 2022



(d) Accuracy on CIC IOT 2022

Fig. 5: Metrics for different kinds of tasks on USTC-2016 and CIC IOT 2022 datasets.

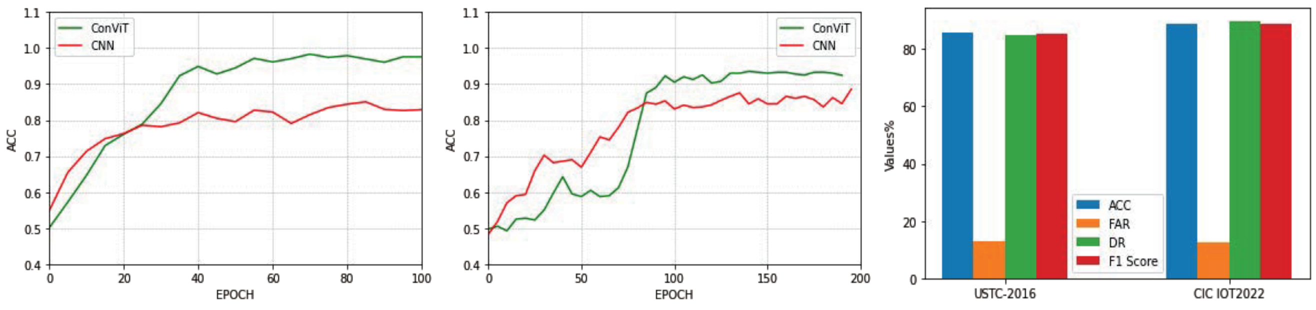
by CNN-LSTM. Both feature extraction models effectively capture both basic and structural features. The performance of GCN is relatively inferior, which may be attributed to the fact that while it extracts contextual information, the basic features of individual packets lack sufficient discriminative power. Moreover, it's worth noting that for USTC-2016, the performance of all models on the CIC IOT 2022 dataset declined, mainly due to the limited categories in CIC IOT 2022.

Moreover, to address the practical deployment requirements in edge environments, we compared the model sizes of the aforementioned networks. The experimental results are presented in TableVII. Notably, our model size is merely 2.45MB, ranking second only to RelationNet, which highlights its exceptional memory efficiency. In stark contrast, CNN-LSTM occupies a size of approximately 4806MB, making it about 1957 times larger than our model.

Training time and inference time are also two critical metrics as users in edge environments require low latency. To compare different network traffic classification models, we select several existing models and classic machine learning approaches. Since ConViTML takes tasks as input, and each task only contains 15 or 19 query samples, we use the average single-sample inference time as an indicator. The results are presented in Table VI, which shows the training time and inference time of various methods on the USTC-2016 dataset. Notably, the ConViTML framework exhibits the lowest training time, taking only 143 seconds, while its inference time ranks second, following only Random Forest and LeNet.

VII. CONCLUSION

In this paper, we propose an end-to-end network traffic classification framework called ConViML, which is suitable for the edge environment and uses DL to automatically extract features from raw traffic files without cumbersome manual operations. For emerging traffic classes, the framework can avoid a lot of resource consumption caused by model updates.



(a) Ablation Experiment on USTC-2016 (b) Ablation Experiment on CIC IOT 2022 (c) Optimal Metrics

Fig. 6: Ablation Experiment on USTC-2016 and CIC IOT 2022 datasets

TABLE VI: Comparison of detection results of the proposed framework with baseline networks

Methods	Metrics	USTC-2016	CIC IOT 2022
RelationNet	ACC(%)	83.75	82.75
	DR(%)	85.50	84.50
	FAR(%)	17.50	19.00
	F1 Score(%)	83.95	83.04
GCN	ACC(%)	66.01	58.00
	DR(%)	66.50	52.00
	FAR(%)	34.52	36.00
	F1 Score(%)	66.16	55.31
ViT	ACC(%)	73.75	67.00
	DR(%)	71.00	68.50
	FAR(%)	23.50	34.50
	F1 Score(%)	73.00	67.48
CNN-LSTM	ACC(%)	94.03	89.66
	DR(%)	97.31	89.16
	FAR(%)	6.32	9.83
	F1 Score(%)	94.37	89.61
ConViTML	ACC(%)	99.75	93.54
	DR(%)	99.66	97.04
	FAR(%)	0.16	1.00
	F1 Score(%)	99.74	98.02

TABLE VII: Comparison of model size of the proposed framework with baseline networks

Methods	Model Size
RelationNet	1.97MB
GCN	18.36MB
ViT	30.12MB
CNN-LSTM	4806.7MB
ConViTML	2.45MB

TABLE VIII: Comparison of training time and inference time of the proposed framework with related research works

Methods	Training time	Inference time
LeNet	201s	$8.9 \times 1e^{-4}$s
GoogleNet	613s	$1.1 \times 1e^{-3}$s
HAST-II	542s	$1.2 \times 1e^{-3}$s
DFR	397s	$1.03 \times 1e^{-3}$s
KNN	-	$3.34 \times 1e^{-2}$s
RandomForest	145s	$1.62 \times 1e^{-5}$s
ConViTML	143s	$9.6 \times 1e^{-4}$s

Specifically, we first split the raw traffic files by session, and then convert each packet into a fixed-format grayscale image as input to our model. Then, we utilize a GAE to extract low-dimensional flow structural features to improve the discriminativeness of features. In addition, the framework leverages meta-learning to enable out-of-distribution traffic sample classification. Finally, we conduct extensive experiments on two real network traffic datasets, and the experimental results show that the ConViML framework can achieve model training and sample classification faster than existing models. Additionally, the training dataset of network traffic collected on edge devices may contain sensitive information that users may not want to upload. In future work, we are interested in introducing federated learning to explore the scalability of ConViML in a distributed architecture. This enables edge devices to collaboratively train a global model without the need to upload training data collected on the devices.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62272069, 62202073, 62072064), Fundamental Research Funds for Central Universities of China (2023CDJYGRH-ZD06, 2021CDJXDJH006), China Postdoctoral Science Foundation (2021M700587), and Chongqing Natural Science Foundation for Postdoctoral (cstc2021jcyj-bshX0077).

References

[1] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1962–1976, 2021.

[2] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, 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.

[3] B. Molina-Coronado, U. Mori, A. Mendiburu, and J. Miguel-Alonso, "Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2451–2479, 2020.

[4] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, 2018.

[5] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile iot networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2410–2422, 2020.

[6] J. Plachy, Z. Becvar, E. C. Strinati, and N. di Pietro, "Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2089–2106, 2021.

[7] L. Nie, X. Wang, S. Wang, Z. Ning, M. S. Obaidat, B. Sadoun, and S. Li, "Network traffic prediction in industrial internet of things backbone networks: A multitask learning mechanism," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7123–7132, 2021.

[8] S. M. Rachmawati, D.-S. Kim, and J.-M. Lee, "Machine learning algorithm in network traffic classification," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 1010–1013.

[9] N. Weng, L. Vespa, and B. Soewito, "Deep packet pre-filtering and finite state encoding for adaptive intrusion detection system," *Computer Networks*, vol. 55, no. 8, pp. 1648–1661, 2011.

[10] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[11] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, 2019.

[12] O. Aouedi, K. Piamrat, and B. Parrein, "Ensemble-based deep learning model for network traffic classification," *IEEE Transactions on Network and Service Management*, 2022.

[13] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

[14] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-stage optimized machine learning framework for network intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1803–1816, 2020.

[15] Y. MALEH and A. Ezzati, "Lightweight intrusion detection scheme for wireless sensor networks." *IAENG International Journal of Computer Science*, vol. 42, no. 4, 2015.

[16] N. Gupta, V. Jindal, and P. Bedi, "Cse-ids: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems," *Computers & Security*, vol. 112, p. 102499, 2022.

[17] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network diversity: a security metric for evaluating the resilience of networks against zero-day attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.

[18] C. Rong, G. Gou, C. Hou, Z. Li, G. Xiong, and L. Guo, "Umvd-fsl: Unseen malware variants detection using few-shot learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[19] J. Guo, X. Zhu, C. Zhao, D. Cao, Z. Lei, and S. Z. Li, "Learning meta face recognition in unseen domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6163–6172.

[20] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[21] 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)*. IEEE, 2017, pp. 712–717.

[22] H. Yang, "Aligraph: A comprehensive graph neural network platform," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 3165–3166.

[23] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE access*, vol. 6, pp. 1792–1806, 2017.

[24] R. Yao, N. Wang, Z. Liu, P. Chen, and X. Sheng, "Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion cnn-lstm-based approach," *Sensors*, vol. 21, no. 2, p. 626, 2021.

[25] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy, "The software suite as a tool for system and network administrators," in *15th Systems Administration Conference (LISA 2001)*, 2001.

[26] Y. Liu, W. Li, and Y. Li, "Network traffic classification using k-means clustering," in *Second international multi-symposiums on computer and computational sciences (IMSCCS 2007)*. IEEE, 2007, pp. 360–365.

[27] S. Dong, "Multi class svm algorithm with active learning for network traffic classification," *Expert Systems with Applications*, vol. 176, p. 114885, 2021.

[28] K. Goeschel, "Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive bayes for off-line analysis," in *SoutheastCon 2016*. IEEE, 2016, pp. 1–6.

[29] Z. Chen, H. Wang, A. Abraham, C. Grosan, B. Yang, Y. Chen, and L. Wang, "Improving neural network classification using further division of recognition space," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 2, pp. 301–310, 2009.

[30] A. J. Malik, W. Shahzad, and F. A. Khan, "Network intrusion detection using hybrid binary pso and random forests algorithm," *Security and Communication Networks*, vol. 8, no. 16, pp. 2646–2660, 2015.

[31] P. Kuttranont, K. Boonprakob, C. Phaudphut, S. Permpol, P. Aimtongkhamand, U. KoKaew, B. Waikham, and C. So-In, "Parallel knn and neighborhood classification implementations on gpu for network intrusion detection," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2-2, pp. 29–33, 2017.

[32] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, vol. 24, no. 11, pp. 1–10, 2015.

[33] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[34] L. Yong and Z. Bo, "An intrusion detection model based on multi-scale cnn," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2019, pp. 214–218.

[35] 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.

[36] Y. Lin, J. Wang, Y. Tu, L. Chen, and Z. Dou, "Time-related network intrusion detection model: a deep learning method," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[37] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "Xai meets mobile traffic classification: Understanding and

improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.

[38] T.-L. Huoh, Y. Luo, P. Li, and T. Zhang, "Flow-based encrypted network traffic classification with graph neural networks," *IEEE Transactions on Network and Service Management*, 2022.

[39] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[40] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," *arXiv preprint arXiv:1807.05960*, 2018.

[41] O. Aouedi, K. Piamrat, G. Muller, and K. Singh, "Federated semi-supervised learning for attack detection in industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2022.

[42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[43] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, and R. Girshick, "Early convolutions help transformers see better," *Advances in neural information processing systems*, vol. 34, pp. 30 392–30 400, 2021.

[44] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the development of a realistic multi-dimensional iot profiling dataset," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*. IEEE, 2022, pp. 1–11.

[45] J. Chen, X.-M. Wu, Y. Li, Q. Li, L.-M. Zhan, and F.-l. Chung, "A closer look at the training strategy for modern meta-learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 396–406, 2020.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[47] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$deep-full-range$: a deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45 182–45 190, 2019.