DFE: Deep Flow Embedding for Robust Network Traffic Classification

Zhijiong Wang, Anguo Zhang, *Senior Member, IEEE*, Hung Chun Li, Yadong Yin, *Member, IEEE*, Wei Chen, *Senior Member, IEEE*, Chan Tong Lam, *Senior Member, IEEE*, Peng Un Mak, *Senior Member, IEEE*, Mang I Vai, *Senior Member, IEEE*, Yueming Gao, *Senior Member, IEEE*, and Sio Hang Pun, *Senior Member, IEEE*

***Abstract*—People's increasing demand for high-quality network services has prompted the continuous attention and development of network traffic classification (NTC). In recent years, deep flow inspection (DFI) is considered to be the most effective and promising method to solve the NTC. However, DFI still cannot effectively address the problem of changes in flow characteristics of complex packet flows and the discovery of new traffic categories. In this paper, we propose a metric learning based deep learning solution with feature compressor, named deep flow embedding (DFE). The feature compressor is used to compress the feature information transmitted layer by layer in DL backbone while maintaining the computational accuracy, so that the backbone can remove as much noise, redundancy, and other irrelevant information from the input data as possible, and achieve more robust feature extraction of network traffic flow. The deep learning (DL) backbone generates an embedding vector for each network packet flow. Then the embedding vector is compared with the vector template preset for each traffic type in the template library to determine the category of the packet flow. Experimental results verify that our method is more effective than the traditional DFI methods in overcoming the problems of flow characteristics variation and new category discovery.**

***Index Terms*—Network traffic classification, deep flow embedding, deep learning, feature compression, metric learning.**

Received 15 February 2024; revised 14 October 2024; accepted 22 January 2025. Date of publication 28 January 2025; date of current version 25 April 2025. This work was supported in part by Science and Technology Development Fund, Macau SAR, under Grant 0010/2024/RIC and Grant 004/2023/SKL, in part by the National Natural Science Foundation of China under Grant 62306001, in part by China Postdoctoral Science Foundation under Grant 2024M750007, in part by the National Key R&D Program under Grant 2022YFE0115500, in part by the National Natural Science Foundation of China under Grant 62371136, in part by the Project of S&T Department of Fujian Province under Grant 2023J01401, in part by ZUMRI-Lingyange Semiconductor Joint Lab under Grant CP-031-2022, in part by the Lingyange Semiconductor Incorporated, Zhuhai, under Grant CP-017-2022, and in part by Blue Ocean Smart System (Nanjing) Limited under Grant CP-003-2023. Recommended for acceptance by Dr. Dusit Niyato. *(Zhijiong Wang and Anguo Zhang contributed equally to this work.) (Corresponding author: Anguo Zhang.)*

Zhijiong Wang and Hung Chun Li are with the Zhuhai UM Science and Technology Research Institute-Lingyange Semiconductor Incorporated Joint Laboratory, Zhuhai 519031, China, and also with the Lingyange Semiconductor Inc., Zhuhai 519030, China.

Anguo Zhang is with the State Key Laboratory of Analog and Mixed-Signal VLSI, Institute of Microelectronics, University of Macau, Macau 999078, China, and also with the Zhuhai UM Science and Technology Research Institute-Lingyange Semiconductor Incorporated Joint Laboratory, Zhuhai 519031, China (e-mail: anguo.zhang@hotmail.com).

Yadong Yin is with the School of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, China.

Wei Chen is with the Center for Intelligent Medical Electronics, School of Information Science and Technology, and Human Phenome Institute, Fudan University, Shanghai 200437, China, and also with the Human Phenome Institute, Fudan University, Shanghai 200437, China.

Chan Tong Lam is with the Faculty of Applied Sciences, Macau Polytechnic University, Macau 999078, China.

Peng Un Mak is with the Zhuhai UM Science and Technology Research Institute-Lingyange Semiconductor Incorporated Joint Laboratory, Zhuhai 519031, China, and also with the Department of Electrical and Computer Engineering, Faculty of Science and Technology, University of Macau, Macau 999078, China.

Mang I Vai and Sio Hang Pun are with the State Key Laboratory of Analog and Mixed-Signal VLSI, Institute of Microelectronics, University of Macau, Macau 999078, China, also with the Zhuhai UM Science and Technology Research Institute-Lingyange Semiconductor Incorporated Joint Laboratory, Zhuhai 519031, China, and also with the Department of Electrical and Computer Engineering, Faculty of Science and Technology, University of Macau, Macau 999078, China.

Yueming Gao is with the School of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, China, and also with the Key Lab of Medical Instrumentation and Pharmaceutical Technology of Fujian Province, Fuzhou University, Fuzhou 350108, China.

Digital Object Identifier 10.1109/TNSE.2025.3535577

I. Introduction

IN CONTEMPORARY network management, network traffic classification (NTC) is vital for quality of service (QoS) initiatives [1], [2], [3], network planning [4], [5], [6], and ensuring security [7], [8], [9], [10]. As new network applications proliferate, traffic patterns diversify, and encrypted traffic becomes increasingly common, these developments pose significant impediments to traditional NTC techniques. Port-based and packet inspection methods, previously effective, falter in the face of such advancements, necessitating the exploration of enhanced NTC methodologies.

Conventional approaches to NTC, such as port-based [11], [12], payload-based [13], [14], statistics-based [15], [16], [17], and behavior-based [18], [19] methods, have achieved commendable performance levels. Nevertheless, the modern tendency of dynamic port selection and payload encryption significantly limit the efficacy of port and payload methods [20], [21]. Besides, although statistical and behavioral techniques offer some insights into encrypted traffic, they are constrained by their labor-intensive feature engineering processes, complicating long-term operations and scalability [22], [23], [24], [25].

With these challenges, the focus has shifted toward Machine Learning (ML) and Deep Learning (DL) given their proficiency in analogous pattern recognition domains like image and speech analysis [26], [27]. However, DL models are not without drawbacks, as they depend on extensive labeled datasets, are sensitive

TABLE I
FEATURE COMPARISON BETWEEN THE PREVIOUS NTC METHODS AND OUR PROPOSED DFE

Feature	Port-Based Methods	Deep Packet Inspection	Deep Flow Inspection	Behavior-Based Methods	Deep Flow Embedding (ours)
Instant classification	●	○	○	○	○
Dynamic port number utilization	○	●	●	●	●
Encrypted data	○	○	●	●	●
Cluster similar applications	○	○	●	●	●
New traffic class	○	○	○	○	●
Evolution of protocol characteristics	○	○	○	○	●
Robustness to noise and interference	●	○	○	○	●

* ○: without this feature; ●: with this feature.

to shifts in traffic distribution, and can be impeded by noisy data [28], [29], [30]. The obsolescence of these models is rapid in dynamic traffic scenarios due to their need for frequent retraining [31], [32]. Thus, to address the issue of excessive retraining overhead, methods rooted in incremental learning have been introduced [33], [34], [35]. These methods are designed to integrate new applications or services into pre-trained deep learning-based traffic classifiers without the need for a comprehensive retraining, thereby accelerating the update cycle of the model. Yet, the feature extraction backbone of these methods is often too simplistic, lacking the robust computing power required, making it challenging to meet the demands of complex scenarios and intricate classification NTC tasks.

Addressing the outlined limitations, this paper presents a deep flow embedding (DFE) framework for NTC, which is inspired by the advancements in DL based metric learning scheme [36], [37]. To more effectively contend with the prevalent issue of noise in network traffic features, we propose an intricately designed feature compressor tailored for integration with the DL backbone. This feature compressor measures the feature information within different layers in DL backbone and strategically incorporates it into the training loss function. Such an approach facilitates the development of a high-accuracy DFE model for NTC endeavors while ensuring minimal loss of feature information. Our innovative design markedly improves the DL model's capacity to discern and exclude noise, redundancy, and interference, substantially enhancing model robustness. Departing from the conventional DL paradigms, our DFE framework outputs an intermediate representation as high-dimensional numerical vector instead of undertaking categorical data classification. We maintain a repository of baseline feature vectors, each emblematic of a given network traffic type, within our model's parameters. For every iteration of incoming traffic data, the DFE model calculates L2-norm distances to ascertain similarity with the stored baselines. Should this distance fall below a predetermined threshold, it indicates that the incoming sample aligns with the traffic type of its respective baseline, thus affirming its classification with exceptional accuracy.

For network flows undergoing protocol changes or featuring new types of traffic, it suffices to catalog their feature vectors—computed by the DFE network—in the appropriate category within the template library, eliminating the need for data collection and retraining of deployed DFE networks. This innovative approach significantly enhances the adaptability and robustness of DL across various network environments and evolving communication protocols while simplifying NTC task maintenance.

In Table I, we benchmark the unique strengths of our DFE framework against three traditional NTC methods across six metrics. The findings illuminate that only the port-based and deep packet inspection methods hold an edge in instantaneous classification and dynamic port number utilization, respectively. In contrast, deep flow inspection (DFI) and behavior-based techniques are adept at decrypting data and grouping similar applications. Though sharing the common challenge of limited instant classification with other machine learning or deep learning NTC methods, DFE proves to be a formidable solution for the remaining five issues: dynamic port utilization, encrypted data processing, clustering similar applications, recognizing novel traffic classes, and adapting to protocol evolution. Empowered by our tailored Feature Compressor, DFE demonstrates superior anti-noise robustness.

The main contributions of this paper are as follows:

- *New Traffic Type Adaptation:* The proposed DFE framework effectively adapts to new traffic types without needing extensive retraining. When a new network traffic category emerges or an existing category changes, the framework can update or add to the feature template library, ensuring continuous adaptability.
- *Feature Compression:* The DFE framework employs a feature compressor within the deep learning backbone, compressing information across multiple layers to remove noise, redundancy, and irrelevant data. This enhances the robustness of feature extraction while maintaining computational accuracy.
- *Metric Learning-Based Model Training:* The framework utilizes a metric learning approach involving triplet loss to train the model. This allows it to accurately classify different types of traffic by generating distinct embedding vectors for each category stored in a template library.
- *Comprehensive Validation Experiments:* Extensive experimental evaluations, including comparisons with other deep learning methods, demonstrate the superior performance of DFE. The framework achieves significant improvements in both precision and recall metrics on different datasets, highlighting its state-of-the-art capability.

The remainder of the paper is structured thusly: Section II delineates the traditional NTC methodologies, which span port-based, payload-based, statistics-based, and behavior-based techniques. Section III delineates the deep flow embedding framework and the construction of the DFE framework model. Section IV details the experimental setup, including the dataset, selected features, and the benchmarks for evaluation. Section V contrasts our results with cutting-edge alternatives. Finally, Section VI synthesizes our findings and possible future work into a conclusion.

II. Brief Review of Network Traffic Classification

A. Background

1) Port-Based Inspection: In the early stage of Internet, most network services follow the port allocation protocol formulated by IANA (the Internet Assigned Numbers Authority) [38], so the type of service can be easily identified through port detection. The main advantages of this technique are low complexity, easy understanding, fast classification and simple implementation. However, nowadays, more and more network applications begin to use self-defined or dynamic port numbers, making port-based traffic identification technology no longer useful. For example, in [39], the authors observed that the accuracy rate is less than 70% if using the port number to identify the traffic class according to the IANA port allocation list.

2) Payload-Based Inspection: Payload-based inspection, also known as deep packet inspection (DPI), as the second generation of traffic classification methods, were proposed to overcome the limitation of port-based techniques. DPI methods classify traffic by inspecting packet headers and payloads with pre-defined signatures of network protocols [40], [41], [42], [43], [44], [45]. DPI methods can deal with dynamic port numbers and provide high-accuracy identification. However, DPI cannot tackle encrypted traffic and needs to continuously maintain the signature matching library up-to-date to identify the evolution of the application [20], [46].

3) Statistics-Based Inspection: Statistics-based inspection uses flow-level statistical characteristics, such as package length, inter-arrival time of packets, flow duration, traffic flow idle time, etc., for traffic classification [47]. The measured characteristics are designed to be unique for each specific type of application, so that can distinguish different applications among others. Classical statistics-based inspection includes both supervised learning methods, e.g., decision tree/forest [48], [49], [50], [51], support vector machine (SVM) [52], [53], naive Bayes [54], and unsupervised/semi-supervised learning methods, e.g., clustering algorithms represented by K-means [55], Gaussian mixture model [56]. DFI can circumvent the problem of payload encryption and privacy security, but due to their handcrafted feature selection, simplicity and incapability of learning complex patterns, classical DFI methods cannot achieve high accuracy and maintain robustness [57].

4) Behavior-Based Classification: The related work in the field of behavior-based network traffic classification has progressed with the integration of deep learning techniques due to their effective feature learning capabilities. While auto-encoders (AEs), convolutional neural networks (CNNs), and long short-term memory (LSTM) networks have been referenced for their foundational importance in deep learning applications, recent advancements and literature have demonstrated more specific contributions and challenges to the field of NTC [58], [59].

In recent years, various adaptations of these deep learning architectures have been proposed to deal with the complexities and characteristics of network traffic data. For instance, modifications to traditional AE architectures have been explored to better capture, the statistical properties of network traffic, addressing issues such as variable traffic volumes and dynamic behavior patterns [60], [61], [62]. These studies have attempted to overcome the limitations in generalization that AEs sometimes exhibit when dealing with highly heterogeneous network environments.

CNNs are inherently apt for handling spatial data and have shown promising results when addressing the classification of traffic that exhibits distinct spatial feature representations. However, recent research has pointed out challenges in CNN's application to network traffic classification, primarily regarding the capture of temporal dependencies within traffic flows [24], [63], [64]. Despite CNNs' ability to handle large-scale data, their effectiveness can be hindered by the lack of sequential processing, which is crucial for understanding the time-based evolution of traffic flows.

LSTMs have been specifically noted for their capacity to model time-series data, which aligns well with the nature of network traffic [65]. The introduction of advanced LSTM variations such as Bidirectional LSTMs and LSTM-based attention mechanisms aim at enhancing the model's ability to recognize long-term dependencies [66], [67], [68]. While these advancements have demonstrated improved performance over traditional LSTMs, they still face limitations in terms of training complexity and computational resources required, which can be prohibitive for real-time network traffic analysis [69], [70].

B. Related Work

Recent studies have proposed various models that combine the strengths of CNNs in feature extraction to form a more robust approach for NTC [76], [77]. These combined architectures, while powerful, are not without issues; they often entail intricate tuning and increased model complexity, which may result in models that are difficult to interpret and operate practically. In [76], a deep learning model that integrates CNN and RNN was presented. The study demonstrated that this method delivers superior detection results compared to alternative algorithms, and notably, it does so without the need for feature engineering, which is typically required by other models. Similarly, [26] proposed a deep learning framework utilizing both CNNs and bidirectional Long Short-Term Memory (BiLSTM) networks to classify both encrypted and non-encrypted traffic, achieving state-of-the-art performance on several benchmark datasets. The App-Net [78] also introduced an end-to-end hybrid neural network that combines RNN and CNN in a parallel configuration to extract effective features from raw TLS flows for mobile app identification. This architecture is capable of learning a joint

TABLE II
HYPOTHETICAL LITERATURE WORKS ON NETWORK TRAFFIC CLASSIFICATION USING DEEP LEARNING

Method	Characteristics	Limitations
Deep Packet [26]	Uses CNNs to classify encrypted network traffic and applies transfer learning.	Requires large labeled datasets; struggles with zero-day attack classification.
Self-Adaptive Network [71]	Online learning that updates the model with new data continuously.	Vulnerable to concept drift and adversarial attacks.
Ensemble Deep Learning [72]	Combines multiple models to improve classification robustness.	Resource-intensive; potential overfitting.
Zero-Shot Learning [73]	Classifies traffic types unseen during training using zero-shot learning.	Lower performance on completely unseen traffic types.
EBSNN [74]	Automatic feature extraction; effective handling of encryption.	Model update requirement.
Graph Neural Network [75]	Employs graph neural networks for network pattern identification.	Computationally expensive; may not scale well to large networks.
Class Incremental Learning [33]–[35]	Integrate new class without a comprehensive retraining	Simple backone with limited computational capability.
DFE (ours)	New traffic type adaptation; no model update requirement; noise tolerance	Manual feature extraction.

flow-app embedding that captures both flow sequence patterns and unique app signatures.

[58] emphasized using deep learning for traffic classification in encrypted environments. Their study demonstrated that deep learning could significantly improve classification accuracy despite the obfuscation introduced by encryption techniques. The Flow Sequence Network (FS-Net) [28] was developed as an end-to-end model that extracts representative features from raw flows of encrypted traffic and classifies them within a unified framework. Additionally, [28] employs a multi-layer encoder-decoder structure to deeply explore the sequential characteristics of flows and incorporates a reconstruction mechanism to enhance the effectiveness of the features. [79] proposed a deep learning classifier for mobile NTC that automatically extracts flow features, operates with encrypted traffic, and identifies complex traffic patterns. In [80], a novel multimodal multitask deep learning approach for NTC is proposed. This model leverages traffic-data heterogeneity by learning both intra- and inter-modality dependencies. It overcomes the performance limitations of existing single-modal deep learning-based classifiers and simultaneously addresses various traffic categorization challenges specific to different providers' needs. Training such models requires a substantial amount of labeled data, while making the labeling process the most challenging and time-consuming. To address this issue, [81] reformulate NTC into a multi-task learning framework, predicting bandwidth requirements and flow duration along with the traffic class. Further, considering the high computational overheads for encrypted NTC problem, [82] proposed MATEC, a lightweight and online approach by leveraging the "Maximizing the reuse of thin modules" design principle, incorporating multi-head attention and convolutional networks. Thanks to the one-step interaction of all packets and parallel computing via the multi-head attention mechanism, MATEC significantly reduces both the number of parameters and the running time.

While traditional deep learning classifiers have set the stage for behavior-based NTC, it is the adaptations and evolution of these methods that are of significant relevance to the current research landscape. Iterative improvements and novel hybrid architectures aim to mitigate the shortcomings of earlier approaches such as feature extraction fidelity [26], [74], temporal pattern recognition [33], [34], [35], and computational efficiency [71], [72], [73], [75]. The recent works have incrementally advanced the state-of-the-art, yet challenges remain, such as improving real-time classification capabilities, reducing computational demands, and enhancing model generalizability across diverse network conditions. A summary of the characteristics and limitations of some typical deep learning methods is as shown in Table II.

III. PROPOSED METHOD

A. Framework

The proposed framework is as showed in Fig. 1, where the "deep learning backbone" is trained by using large-scale labeled dataset. There are three stages during the inference of traffic classification. In the first stage, the deep learning backbone is trained by using the triplet loss of (9) on training data containing different types of network traffic. After completing the training, we use the template data of different types of traffic to go through the primary feature extraction and the high-level feature extraction of the deep learning model to obtain the embedding vector of the template of each traffic category. These embedding vectors are saved in the template library. In the second stage, for the test data, primary feature extraction will also be performed and input into the deep learning model. The generated embedding vector will be compared with the embedding vector in the template library to infer the traffic type of the test data. In the third stage, when the data flow characteristics of a certain service type change (for example, the network protocol is updated, the network transmission medium changes), or a certain type of network service is added, the original feature template library needs to be updated.

B. Metric Learning With Feature Compression

The schematic diagram of triplet training with feature compression is as given in Fig. 2, where the deep learning backbone

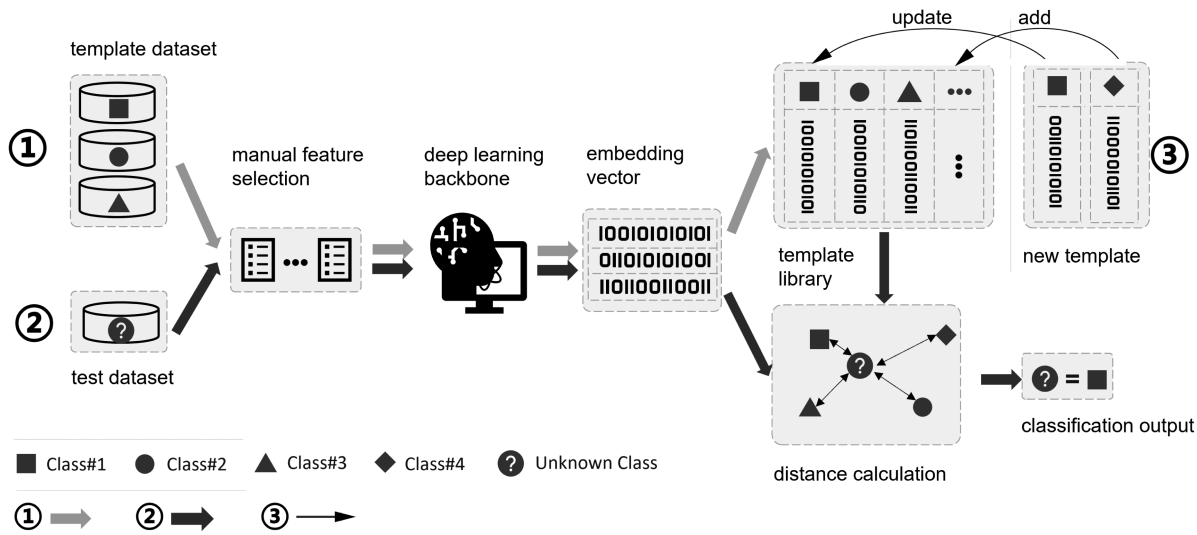


Fig. 1. Proposed framework of "deep flow embedding". The deep learning backbone is trained on the training dataset by using triplet loss. Stage ①: The *template dataset* contains different classes of network traffic data, *manual feature selection* extracts and formats the desired features, and feeds them into the *deep learning backbone* to produce more abstract and higher-level semantic features. Then, these semantic features are encoded into a fixed-length *embedding vector* and stored in the *template library*. Stage ②: The *test dataset* is also processed by the *manual feature selection* and *deep learning* modules, and then output an *embedding vector* for each traffic flow, distance between the output vectors and *template library* are calculated to classify the corresponding traffic flow. Stage ③: If there is a new network traffic category or a network protocol of an existing traffic category is changed, the newly generated traffic template will be used to update or add to the *template library*.

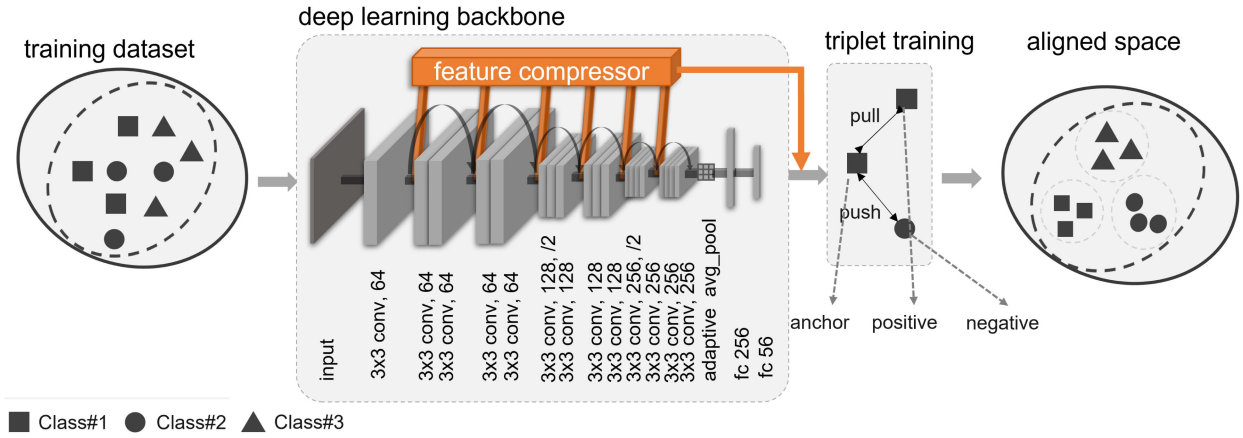


Fig. 2. Schematic diagram of the metric learning with feature compression, different solid shapes (squares, circles, triangles) represent different traffic categories.

consists of an input layer, 13 convolution (conv), an adaptive average pooling layer, 2 fully-connected (fc) layers, residual connections and a feature compressor that receives intermediate features of multiple backbone layers. The notation "3×3 conv, 64" means that the layer has 64 convolutional kernels with size of 3×3. The suffix "/2" represents the parameter *stride* of convolutional kernels is set to 2. The notation "fc 256" means the fully-connected layer has 256 output neurons with ReLU activation function. It should be noted that the model in Fig. 2 is not mandatory, other backbones that meet the design concept can also be used for the construction of the model.

The metric learning with feature compression in Fig. 2 tries to use triplet training to shorten the distance between the output vectors of samples of the same class, and push the distance between samples of different classes. In addition, it also receives compression losses calculated by the feature compressor, trying to achieve layer-by-layer information compression by reducing the amount of information transmitted backwards through the backbone while ensuring identification accuracy. Therefore, our proposed method consists of two major parts, namely feature compression and triplet training, which we will detail below.

1) Feature Compression: In the backbone of Fig. 2, we extract features from a total of 6 layers, starting from the input layer, to compress as much feature information as possible while ensuring network computational accuracy. The aim is to optimize the network's feature extraction ability, allowing for the use of as little feature information as possible to achieve the desired performance. The network compresses the feature information by eliminating any noise, redundancy, or other irrelevant data, ultimately increasing the network's anti-noise robustness.

As seen in Fig. 2, denote the features of 6 layers as $f_i, i = 1, 2, \ldots, 6$, we use mutual information $I(f_i; f_{i+1})$ to quantify the relevance between two feature layers f_i and f_{i+1}, and $H(f_i|f_{i+1})$ to quantify the conditional entropy. Then, define the feature compression loss function by

$$\mathcal{L}_f = H(x) - \beta_0 \cdot H(x|f_1) - \sum_{i=1}^{5} \beta_i \cdot H(f_i|f_{i+1}) \quad (1)$$

where x is the input data. Since $H(x)$ is a constant value that does not change with the optimization of network weights, it can be removed from the loss function of gradient-based training algorithms. For simplicity, let $x = f_0$, then (1) can be rewritten as

$$\mathcal{L}_f = -\sum_{i=0}^{5} \beta_i \cdot H(f_i|f_{i+1}) \quad (2)$$

Considering that $H(f_i|f_{i+1}) = H(f_i) - I(f_i; f_{i+1})$,

$$\tilde{\mathcal{L}}_f = \sum_{i=0}^{5} \beta_i \cdot I(f_i; f_{i+1}) - \sum_{i=0}^{5} \beta_i \cdot H(f_i) \quad (3)$$

In theory, if DFE is trained to ideal performance, although the original input x (or f_0) may contain noise or redundant information due to network environment and user differences, the extracted features $f_i, i = 1, \ldots, 6$ by the deep learning backbone should be relatively stable with minimal fluctuations, thus $H(f_i)$ is close to 0, we have

$$\tilde{\mathcal{L}}_f \leq \sum_{i=0}^{5} \beta_i \cdot I(f_i; f_{i+1}) \quad (4)$$

In keeping with a similar assumption made in [83], [84], wherein every feature f_i is encoded as the sum of a deterministic function $\mu(f_i)$ and Gaussian noise, *i.e.,*

$$f_{i+1} = \mu(f_i) + \varepsilon_{i+1} \quad (5)$$

where $(\varepsilon|_{f_{i+1}}) \sim \mathcal{N}(\mu(f_{i+1}), \Lambda(f_{i+1}))$, where $\Lambda(f_{i+1})$ is the noise covariance realted to the network weights and the input x. Commonly, the feature f_{i+1} can be regards as a mixture of N Gaussian distributed components, thus consistent with that of [83], the mutual information between f_i and f_{i+1} can be calculated as a non-parameteric upper bound,

$$\begin{aligned} &I(f_i; f_{i+1}) \leq \hat{I}(f_i; f_{i+1}) \\ &= -\frac{1}{N} \sum_p \log \frac{1}{N} \sum_q \\ &\times \exp\left\{-D\left[\mathcal{N}\left(\mu(f_i^p), \Lambda(f_i^p)\right) \| \mathcal{N}\left(\mu(f_i^q), \Lambda(f_i^q)\right)\right]\right\} \end{aligned} \quad (6)$$

where $D[\cdot\|\cdot]$ denotes the Kullback–Leibler divergence (KLD) between two d-dimensional Gaussian distribution, formulated as

$$\begin{aligned} &D\left[\mathcal{N}(\mu, \Lambda) \| \mathcal{N}(\tilde{\mu}, \tilde{\Lambda})\right] \\ &= \frac{1}{2}\left[\ln \frac{\det(\tilde{\Lambda})}{\det(\Lambda)} + (\mu - \tilde{\mu})\tilde{\Lambda}^{-1}(\mu - \tilde{\mu}) + \mathrm{tr}(\tilde{\Lambda}^{-1}\Lambda) - d\right] \end{aligned} \quad (7)$$

where $\det(\cdot)$ and $\mathrm{tr}(\cdot)$ are the determinant and trace of a matrix. (6) and (7) bound the mutual information in terms of the pairwise KLD between the mixture Gaussian componented distribution of f_i and f_{i+1}.

The upper bound of feature compression loss function is

$$\mathcal{L}_f \leq \tilde{\mathcal{L}}_f \leq \sum_{i=0}^{5} \beta_i \cdot \hat{I}(f_i; f_{i+1}) \quad (8)$$

Therefore, reducing the upper bound of $\tilde{\mathcal{L}}_f$ can continuously reduce the loss function $\mathcal{L}_f$, thus achieving compression of the features extracted from the deep learning backbone.

2) Triplet Training: The triplet means that the training data is constructed by a triplet tuple, which formed by anchor, positive and negative samples. Given a triplet as $< x_a, x_p, x_n >$, where the anchor x_a and positive x_p are with the same class, the negative x_n is with the different classes, i.e., $y_a = y_p \neq y_n$. Suppose DFE network maps the triplet into a learned feature space, noted by $< \phi_w(x_a), \phi_w(x_p), \phi_w(x_n) >$, where w is the trained weights of DFE network. The original triplet loss aims to minimize the distance to some extent between two inputs $\phi_w(x_a)$ and $\phi_w(x_p)$, while maximize the distance between $\phi_w(x_a)$ and $\phi_w(x_n)$, so that the quantified difference of the pair (x_a, x_p) is much smaller than that of the pair (x_a, x_n) by a predefined margin,

$$\begin{aligned} &d_1(x_a, x_p, x_n, w) \\ &= d\left(\phi_w(x_a), \phi_w(x_p)\right) - d\left(\phi_w(x_a), \phi_w(x_n)\right) \\ &\leq \alpha_1 \end{aligned} \quad (9)$$

where α_1 is the margin to enforce between the positive pair $(\phi_w(x_a), \phi_w(x_p))$ and negative pair $(\phi_w(x_a), \phi_w(x_n))$. $d(\cdot, \cdot)$ is a pre-defined measurement of the distance between two pairs, this paper uses L2-norm to implement, i.e.,

$$\begin{aligned} d\left(\phi_w(x_a), \phi_w(x_p)\right) &= \| \phi_w(x_a) - \phi_w(x_p) \|_2 \\ d\left(\phi_w(x_a), \phi_w(x_n)\right) &= \| \phi_w(x_a) - \phi_w(x_n) \|_2 \end{aligned} \quad (10)$$

Triplet loss encourages the distance of the positive pair to be smaller than that of the negative pair by the margin α_1. The objective function of triplet learning is to optimize the average value of (9) on all the triplets in dataset $\mathcal{D}$.

However, the original triplet loss (9) does not stipulate how close the pair $(\phi_w(x_a), \phi_w(x_p))$ should be, accordingly, sampled instances of the same traffic class may form a cluster with a relatively large intra-class distance in the feature space. Therefore, to solve the problem for higher classification performance, we consider a new term that requires the quantified difference of the pair (x_a, x_p) be smaller than a designed margin α_2,

$$d_2(x_a, x_p, w) = d\left(\phi_w(x_a), \phi_w(x_p)\right) \leq \alpha_2 \quad (11)$$

Combine the intra-class distance (11) as a penalty term with the inter-class distance (9) of original triplet loss, we have the loss function for NTC problem in (12),

$$\mathcal{L}_{\text{NTC}} = \frac{1}{N} \sum_{\{a,p,n\} \in \mathcal{D}} \underbrace{\max\left\{\delta_w\left((x_a, x_p)|(x_a, x_n)\right) + \alpha_1, 0\right\}}_{\text{inter-class constraint}}$$

$$+\eta\frac{1}{N}\sum_{\{a,p,n\}\in\mathcal{D}}\underbrace{\max\left\{d\left(\phi_w(x_a),\phi_w(x_p)\right),\alpha_2\right\}}_{\text{intra-class constraint}} \tag{12}$$

where $\delta_w((x_a,x_p)|(x_a,x_n)) = d(\phi_w(x_a),\phi_w(x_p)) - d(\phi_w(x_a),\phi_w(x_n))$, and η is a coefficient to weight the inter-class and intra-class constraints.

Thus, the total loss for our proposed deep learning with feature compressor model can be summarized by

$$\mathcal{L} = \tilde{\mathcal{L}}_f + \lambda\mathcal{L}_{\text{NTC}} \tag{13}$$

where λ is a designed coefficient used to balance the weight of the two loss terms.

C. Template Matching

Let the template library be $M = \{(t_1,y_1),(t_2,y_2),\ldots,(t_T,y_T)\}$, where $t_i \in \mathcal{M} \subseteq \mathbf{R}^n$ is the embedding vector, $y_i \in \mathcal{Y} = \{c_1,c_2,\ldots,c_P\}$ is the class of template $i,(i = 1,2,\ldots,T)$, and P is the number of traffic classes.

For a given output vector $\phi_w(x)$ of a traffic service to be identified, calculate the L2-norm distance with all the templates in library M, find k templates that are nearest to $\phi_w(x)$, and denote the neighborhood covering these k templates as $\mathcal{N}_k(x)$. Then, regarding the library M, use the majority voting method to determine the class to which x belongs, i.e.,

$$y = \arg\max_{c_j}\sum_{x_i\in\mathcal{N}_k(x)} I(y_i = c_j), i = 1,2,\ldots,T \tag{14}$$

where $I(\cdot)$ is an indicator function that means $I = 1$ if $y_i = c_j$, otherwise $I = 0$.

IV. Experiment Preparation

A. Dataset

To verify the performance of our proposed framework, the following datasets are used:

UNB ISCX Network Traffic (VPN-nonVPN) [85]: The dataset consists of 12 traffic categories, including email, chat, streaming, file transfer, VoIP and P2P, as well as these traffics running over Virtual Private Network (VPN) sessions. The traffic was captured using Wireshark and tcpdump, with a total amount of 28 GB of data. Captured packets were separated into. pcap files which labeled according to the application that produced the packets.

Self-Collected Traffic Dataset: We also collected the self-generated traffic data by using Docker virtual container. In each Docker container, only a single application was installed to generate network flows, and tcpdump toolkit was used to capture the TCP traffic data. In this way, we could ensure that the network traffic flow only comes from a single application with no traffic data confusion. Moreover, we could easily label the captured flow data by traffic type. As shown in Table III, we collected 8 types of network traffic data by running different applications.

Before inputting the information of network traffic flow into the DFE model, we need to perform the necessary preprocessing on the dataset to obtain the features and adapt to the input specifications of DFE. It mainly includes three steps: dataset to flows, flow to features, and features to image.

TABLE III
Description of the Self-Collected Traffic Dataset

Traffic Type	Applications	Amount
email	Outlook, Foxmail, Google mail, etc.	4515
instant message	QQ, Wechat, Skype, etc.	5241
file transfer	Bittorrent, Thunder, Foxy, etc.	4406
web browsing	Weibo, Sohu news, etc.	4378
voice call	QQ voice, Wechat voice, etc.	3928
online meeting	Tecent Meeting, Zoom, etc.	5710
online game	Honor of Kings, Xbox Live, etc.	4099
location service	Baidu map, Google map, Amap, etc.	3687

1) Dataset to Flows: The raw traffic dataset was stored in. pcap format, so we split the data into flows according to the five tuples, i.e., source IP, source port, destination IP, destination port, the protocol of transmission layer.

It should be noted the dataset was captured in a real-world emulation, there existed some irrelevant or noisy packets that should be washed out. In particular, the TCP packets with either SYN, ACK or FIN flags are used for three-way handshaking procedure while establishing or destroying a connection, and the Domain Name System (DNS) packets in the dataset which are used for translating the hostname (URL) to IP address. These packets carry no information with respect to the applications that can be safely discarded.

2) Flow to Features: As the same with [86], the designed features shown in Table IV are used in this paper. For each traffic flow, we need to extract the above-mentioned 81 features, including 26 time-related and 55 time-unrelated features in advance. The choice of handcrafted features relies on expert knowledge and may not fully capture the inherent complexity and latent patterns within the data. However, in this paper, the selection of handcrafted features as an initial attempt has helped us more clearly understand the important characteristics of network traffic during the development of DFE. The transition to automated, end-to-end feature extraction is a direction worth exploring. The automated feature extraction methods can be a future direction to further enhance the performance and adaptability of the DFE model, which will also be explicitly outlined in Section VI.

3) Features to Image: For the total 81 extracted features of each traffic flow, we reshape them to a 9×9 matrix, which can be regarded as an gray image with 9-pixel width and 9-pixel height. And then, we normalize the scale to range [0, 1].

B. Evaluation

To evaluate the performance of our proposed DFE method, the metrics Recall (Rc), Precision (Pr) and F1 score ($F1$) are applied,

$$Rc = \sum_{i=1}^{C}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

TABLE IV
THE FEATURES USED FOR OUR DFE FRAMEWORK

Type	Features	Description
Time-Related	duration	Flow duration
	min_iat, max_iat, mea_iat, low_quartil_iat, median_iat, upp_quartile_iat, iqd_iat, mode_iat, range_iat, stdrob_iat, skew_iat, exc_iat, nfp_pl_iat, ps_iat_histo	Interval time features
	fb_psec, fp_psec	Bytes/Packet numbers per second
	min_act, max_act, mean_act, median_act, std_act, min_idle, max_idle, mean_idle, median_idle, std_idle	Time calculated regarding to the idle-to-active or active-to-idle states
Non-Time-Related	dir, numPktsSnt, numPltsRcvd, numBytesSnt, numBytesRcvd, minks, maxPktSz, avePktSz, pktps, bytps, pitas, bytes	Flow direction, packet an byte counting
	tcpPSeqCnt, tcpSeqSntBytes, tcpSeqFaultCnt, tcpPackCnt, tcpFlwLssAckRcvd-Bytes, tcpAckFaultCnt, tcpInitWinSz, tcpAveWinSz, tcpMinWinSz, tcpMaxWinSz, tcpWinSzDwnCnt, tcpWinSzUpCnt, tcpWinSzChgDitCnt, tcpAggFlags, tcpAggrA-nomaly, tcpAggrOptions, tcpMSS, tcpWS, tcpOptCnt, tcpS-SA/SA-A_Trip, tcpS-SA-A/A-A_RTT, tcpRTTAckTripMin, tcpRTTAckTripMax, tcpRTTAckTripAve, tcpStates	TCP header
	connSrc, connDst, connSrc⇄Dst	Number of connection from source to different servers
	minPl, maxPl, meanPl, lowQuartilePl, meadianPl, uppQuartilePl, iqdPl, modePl, rangePl, stdPl, stdrobPl, skewPl, excPl	Packet length

$$Pr = \sum_{i=1}^{C} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i},$$
$$F1 = 2 \cdot \sum_{i=1}^{C} \frac{Rc_i \cdot Pr_i}{Rc_i + Pr_i} \tag{15}$$

where C represents the number of classes, and $\mathrm{TP}_i, \mathrm{FP}_i$, and FN_i denote the number of true positives, false positives, and false negatives for each class i, respectively.

C. Parameter Settings

We performed all experiments on a computer with a 3.5 GHz Intel Core i7 processor and 32 GB of RAM, running Ubuntu 16.04 LTS. We used the PyTorch 1.4.0 deep learning framework to build and train all models.

For both the VPN-nonVPN and Self-Collected datasets in Table III, we randomly divided the data into training (80%) and test (20%) sets. We trained all models using stochastic gradient descent (SGD) with a constant learning rate of 0.001. The batch size was set to 32 for all experiments, and the number of epochs for each experiment is up to 200 epochs. We used the loss function, as given in (13), for all experiments, and used early stopping to prevent overfitting. The patience parameter for early stopping was set to 10 epochs. For each traffic class template, we randomly selected 20 samples from the training set. After getting the embedded vectors from the well-trained deep learning backbone, we stored these vectors in the template library as a benchmark for subsequent traffic flow classification. The parameters $\beta_i, i = 0, \ldots, 5$ in (3) were set as 0.01, 0.02, 0.03, 0.04 and 0.05, respectively. According to the commonly setting, α_1 and α_2 were fixed by 0.5 and 2.0, respectively.

V. EXPERIMENTAL RESULTS

In our comparative analysis, we benchmarked the performance of our proposed DFE against several commonly-used machine learning and deep learning methods. These include Random Forest (RF), Auto-encoder (AE), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM). For each of these models, we meticulously configured and optimized the hyperparameters to ensure fair and competitive comparisons.

RF: We utilized a Random Forest classifier with 100 trees. The maximum depth of the trees was set to 20, and we used the Gini impurity function as a criterion. Feature subset selection for each tree was performed by randomly selecting sqrt (number of features).

AE: For the AE, we designed an auto-encoder with three layers. The encoding layer reduces the feature dimension to 128, followed by a latent representation layer of 64 units, and finally, a decoding layer that restores the dimension to the original feature size. We used ReLU activation functions and a learning rate of 0.001.

CNN: Our CNN architecture consisted of two convolutional layers with 32 and 64 filters, respectively, each followed by max-pooling layers. A fully connected layer with 128 units preceded the output layer. We used a stride of 1, a pool size of 2, and ReLU activations across the network.

LSTM: The LSTM model was configured with two layers, each containing 50 units. To combat overfitting, a dropout of 0.5 was implemented after each LSTM layer. The models were trained with a batch size of 64 and a learning rate of 0.01.

For the above two datasets, we first trained the models on the respective training sets, and then tested their performance in terms of Recall Rc and Precision Pr on the test sets. Further, we also used the traffic generated by new applications to

TABLE V
PERFORMANCE COMPARISON BETWEEN THE PREVIOUS NTC METHODS AND OUR PROPOSED DFE

Model	Metric	Train: VPN-nonVPN training set; Test: VPN-nonVPN test set											
		email	chat	streaming	file transfer	VoIP	P2P	VPN-email	VPN-chat	VPN-streaming	VPN-file transfer	VPN-VoIP	VPN-P2P
RF	Pr [%]	82.5	71.2	86.8	96.3	87.2	98.4	96.4	98.6	93.8	92.4	95.6	97.6
	Rc [%]	85.4	83.4	92.0	98.2	71.1	97.2	95.2	95.0	94.9	89.6	87.3	95.4
	$F1$ [%]	83.9	76.8	89.3	97.2	78.3	97.8	95.8	96.8	94.4	91.0	91.3	96.5
AE	Pr [%]	84.1	70.7	85.4	97.8	88.4	97.5	94.1	97.1	92.1	91.9	94.2	97.0
	Rc [%]	80.9	86.1	92.4	98.4	65.9	99.0	95.0	97.6	93.7	90.3	82.1	92.9
	$F1$ [%]	82.5	77.6	88.8	98.1	75.5	98.2	94.5	97.3	92.9	91.1	87.7	94.9
CNN	Pr [%]	91.2	75.7	88.1	**98.4**	**90.4**	99.2	97.9	98.2	95.4	93.0	96.7	98.5
	Rc [%]	89.6	**89.3**	93.3	**99.6**	79.5	**99.6**	98.2	98.6	95.3	94.6	89.5	97.4
	$F1$ [%]	90.4	81.9	90.6	**99.0**	84.6	99.4	98.0	98.4	95.4	93.8	93.0	97.9
LSTM	Pr [%]	86.0	**85.6**	87.5	97.9	86.5	98.0	96.7	**99.5**	93.1	91.7	94.5	97.1
	Rc [%]	82.9	82.0	92.8	98.7	76.0	98.2	97.3	99.1	94.6	92.2	**90.8**	95.9
	$F1$ [%]	84.4	83.8	90.1	98.3	80.9	98.1	97.0	99.3	93.8	91.9	92.6	96.5
DFE (ours)	Pr [%]	**91.9**	84.4	**89.3**	98.2	89.8	**99.3**	**98.4**	99.4	**96.8**	**94.3**	**97.8**	**99.7**
	Rc [%]	**92.3**	87.2	**93.5**	99.5	**82.6**	**99.6**	**99.7**	**99.8**	**96.4**	**95.7**	89.4	**98.9**
	$F1$ [%]	**92.1**	**85.8**	**91.4**	98.8	**86.0**	**99.4**	**99.0**	**99.6**	**96.6**	**95.0**	**93.4**	**99.3**

* RF: random forest; AE: auto-encoder; CNN: convolutional neural network; LSTM: long short-term memory.

perform the NTC task. These applications had not been used in the original two datasets. For some network traffic types, the flow characteristics generated by the new applications may be quite different from those of the original dataset, so that the classification performance of the compared four machine learning models may be degraded by a certain extent. The four machine learning models require retraining for this problem, however, our DFE adopts a triple-loss based metric learning scheme, and thus has the advantage of not requiring retraining for new classes of data, it only needs to generate template vectors for the newly generated traffic flows. Therefore, we also compared the classification performance of template vectors without and with the traffic flows added of new applications. Each model was cross-validated using 5-fold cross-validation to ensure the stability and reproducibility of the performance results.

A. Results on VPN-NonVPN Dataset

Table V shows the numerical comparison of the classification performance of our proposed DFE and four other machine learning models on the original VPN-nonVPN dataset. Among the total 12 traffic categories, our DFE achieves the best classification precision Pr and the best recall rate Rc in nine of them. Regarding the F1 score, DFE outperforms other methods except for the file transfer traffic. While even in the file transfer traffic, DFE is only 0.2% behind the optimal CNN method, indicating that this performance gap is small to be negligible. The Pr is only lower by 1.2%, 0.6% and 0.1% compared to the best performance on chat, VoIP and VPN-Chat, while Rc is only lower by 2.1%, 0.1% and 1.4% compared to the best performance on chat, file-transfer and VPN-VoIP. On the whole, it can be seen that the DFE method has achieved the best classification results in terms of metrics Pr and Rc in most traffic types, and the CNN also achieves performance that is not much different from our DFE. This is due to the use of convolution operations in our DFE and CNN models to perform further deep feature extraction on the generated traffic flow feature images. The "local region connection, translation invariance" features of convolution operations can better improve anti-noise robustness and generalization, while the "weight sharing, pooling (down sampling)" feature can reduce the number of model parameters and expand the model receptive field, thereby effectively avoiding overfitting and improving the computational accuracy.

TABLE VI
PERFORMANCE COMPARISON BETWEEN OUR DFE AND OTHER COMMON MACHINE LEARNING METHODS ON VPN-NONVPN DATASET

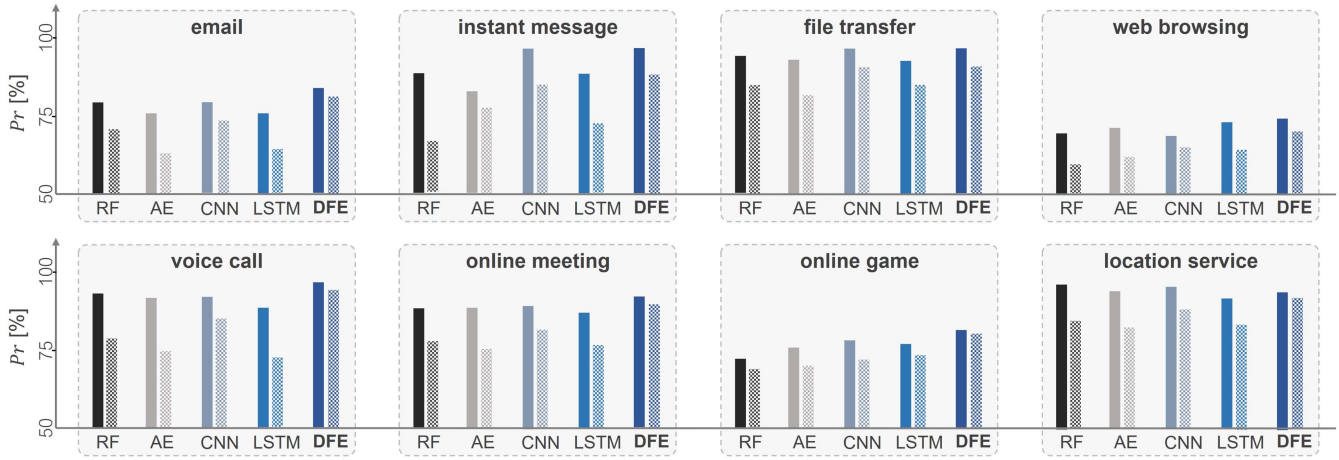
Paper	Method	Precision [%]	Recall [%]
Lotfollahi *et.al* [26]	CNN	94.0	93.0
Bu *et.al* [64]	Network-in-Network	97.9	97.9
Chiu *et.al* [27]	Convolutional AE	97.36	97.33
Telikani *et. al* [25]	Stacked AE	97.1	91.1
Telikani *et. al* [25]	CNN	98.4	92.6
Chen *et.al* [87]	CNN + Metric Learning	98.53	98.57
Ren *et.al* [46]	Tree-RNN	98.98	98.97
our DFE	Deep Feature Embedding	**99.21**	**99.16**

We also compared our DFE with other deep learning-based methods, as shown in Table VI, compared with the CNN classifier [26], Network in Network [64], CNN-based metric learning [87] and Tree-RNN [46], DFE obtains 5.21%, 1.31%, 0.68% and 0.23% improvement in precision Pr metric, and 6.16%, 1.26%, 0.59% and 0.19% in recall rate Rc, respectively. It suggests that the proposed DFE achieved a SOTA performance on Pr and Rc indicators.

TABLE VII
PERFORMANCE COMPARISON BETWEEN OUR DFE AND OTHER COMMON MACHINE LEARNING METHODS, WHERE ALL THE MODELS ARE TRAINED ON VPN-NONVPN DATASET, WHILE TESTED ON OUR SELF-COLLECTED TEST SET OF NEW APPLICATIONS

Model	Metric	Train: VPN-nonVPN training set; Test: Self-Collected test set											
		email	chat	streaming	file transfer	VoIP	P2P	VPN-email	VPN-chat	VPN-streaming	VPN-file transfer	VPN-VoIP	VPN-P2P
RF	Pr [%]	69.0	58.0	75.5	84.7	72.2	86.9	82.6	86.3	83.3	79.6	83.5	84.7
	Rc [%]	74.8	72.8	77.5	87.3	59.0	84.5	83.0	83.7	80.7	76.2	76.3	83.1
AE	Pr [%]	73.4	60.0	71.8	84.6	75.2	83.3	81.9	83.2	81.2	81.7	82.9	85.7
	Rc [%]	70.4	75.6	79.6	83.4	55.1	86.0	84.3	83.8	82.8	76.2	71.9	79.1
CNN	Pr [%]	81.2	64.9	77.1	87.5	78.4	87.5	87.7	84.4	82.0	79.2	82.0	87.3
	Rc [%]	77.5	78.4	82.2	88.3	68.6	88.1	86.7	84.8	80.8	83.9	76.2	87.0
LSTM	Pr [%]	72.7	74.6	77.1	85.9	72.7	85.7	85.1	88.9	80.5	79.0	79.8	83.2
	Rc [%]	69.3	70.4	78.2	88.3	61.6	86.0	84.0	85.6	81.0	80.5	79.9	82.5
DFE-	Pr [%]	79.2	72.8	75.7	84.7	78.0	87.5	83.6	87.0	86.0	81.5	83.1	86.1
before	Rc [%]	81.8	76.1	80.7	87.4	69.1	86.8	85.0	88.7	81.6	83.7	75.4	85.6
DFE-	Pr [%]	**90.1**	**80.7**	**87.8**	**91.5**	**87.6**	**93.4**	**95.1**	**92.7**	**91.5**	**92.2**	**91.9**	**93.6**
after	Rc [%]	**87.5**	**86.8**	**91.9**	**93.2**	**79.4**	**96.5**	**95.2**	**94.8**	**89.9**	**86.7**	**87.2**	**92.3**

* RF: random forest; AE: auto-encoder; CNN: convolutional neural network; LSTM: long short-term memory.



* RF: random forest; AE: auto-encoder; CNN: convolutional neural network; LSTM: long short-term memory.
* Solid bar: the traffic data of training set and test set is captured from the same applications (test set A); Dotted bar: the traffic data is captured from different applications (test set B).

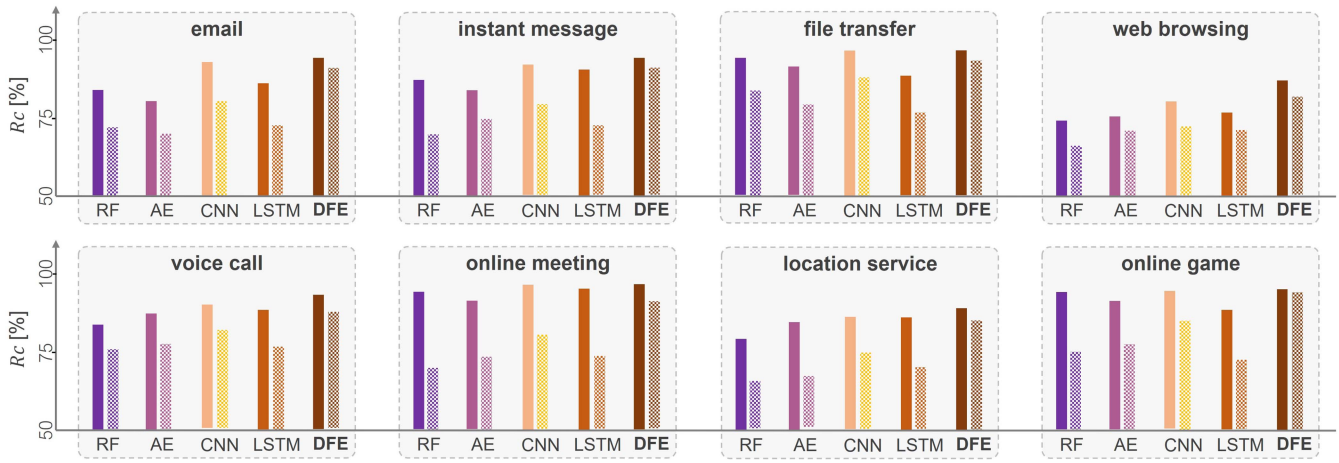
Fig. 3. Comparison of classification precision Pr on self-collected dataset.

B. Results on Self-Collected Dataset

For the self-collected traffic data, we made a training set and two test sets A and B. The data used in test set A and training set are both captured from the same applications, while test set B is captured from applications that different from the training set.

In Table VII, we presented the classification performance on the traffic datasets generated by other new applications, where all the machine learning models are trained on the VPN-nonVPN dataset. DFE-before, DFE-after denote the template vectors of new applications have not been and have been added and has been added to the DFE model, respectively. It can be seen that for the new traffic dataset, all models show some degree of accuracy drop. Before adding the template vectors of new applications, our DFE-before and CNN achieve relatively higher classification performance, which also proves that the deep learning-based models have higher computational capability. Both the metrics Pr and Rc of our DFE-after are significantly improved by simply adding template vectors.

In Figs. 3 and 4, the classification performance of several methods on test sets A and B are shown, respectively. On test set B, we pre-extracted some embedding templates of the traffic flow of new applications and put them into the template library of DFE. It can be seen that compared with the test set A, the performance of different methods on the test set B has decreased, because the characteristics of traffic flow generated by new applications may be different from those in the training set, while these differences may cause these methods to produce incorrect recognition results. For test set A, except for the classification



* RF: random forest; AE: auto-encoder; CNN: convolutional neural network; LSTM: long short-term memory.
* Solid bar: the traffic data of training set and test set is captured from the same applications (test set A); Dotted bar: the traffic data is captured from different applications (test set B).

Fig. 4. Comparison of recall rate Rc on self-collected dataset.

TABLE VIII
PERFORMANCE COMPARISON BETWEEN OUR DFE AND OTHER NETWORK TRAFFIC EMBEDDING METHODS ON SELF-COLLECTED DATASET

Paper	Method	Precision [%]	Recall [%]
Cui *et.al* [88]	LSTM + Word Embedding	80.2	82.7
Zhao *et.al* [89]	AE + N-Gram embedding	83.1	82.9
Hyeokmin *et.al* [90]	LSTM + Feature Embedding	86.5	88.3
Handika *et. al* [91]	Embedding Neural Network	88.6	87.9
our DFE	Deep Feature Embedding	**91.2**	**90.4**

precision Pr of location services, our DFE outperforms the other four methods in the remaining 7 network traffics (in instant message and file transfer categories, CNN obtained Pr values similar to DFE). In email, file transfer, online meeting, and online game categories, only CNN can achieve recall rates Rc comparable to DFE. For test set B, DFE is the best in both Pr and Rc, and its performance drop is also the minimal. These results mean that DFE can achieve optimal classification performance on datasets of the same source, while for datasets from different sources, it is only necessary to add several embedding vectors to the template library to achieve less lossy classification performance. Thus, the applicable scenarios of our proposed DFE are enhanced.

Further, in Table VIII, we also presented the performance comparison between our DFE with other network traffic embedding methods on Self-Collected dataset. Compared to other methods, our DFE achieves a 2.6% and 2.1% increase in the highest Precision (Pr) and Recall (Rc) values, respectively. We believe that this is mainly due to two factors. Firstly, we employ CNN as the deep learning backbone to extract features. The weight-sharing property of CNN avoids overfitting, and its convolutional characteristics offer translation invariance for robust feature extraction. Secondly, we propose a feature compressor that efficiently reduce noise, interference, and redundant information to mitigate their adverse effects on feature extraction.

C. Feature Representation

We compared the feature representation of the proposed DFE and other three commonly used neural network models (i.e., AE, LSTM and CNN) for different types of network traffic flows. For AE, LSTM and CNN, we extracted features from the previous layer of softmax classification layer. With respect to the same type of traffic flows, if the model can output the embedding vectors close together, while the embedding vectors of different types are far apart, it means that the model can well extract the specific and essential features of the traffic flow, as well as effectively remove noise or redundant data, so as to achieve high-precision and strong-robust computing performance. As shown in Fig. 5, we performed principal component analysis (PCA) on the traffic flow features that extracted by different network models, and perform a 2D plane projection on the first two principal components. It can be clearly seen the proposed DFE can obtain a more aggregated feature distribution of the same type of traffic flow, and that of different types is sparse. Therefore, the clustering results in Fig. 5 also correspond with the DFE's optimal NTC accuracy in the previous experimental results, and provide more space for the discovery of new unknown traffic types.

D. Parameter Exploration

Figs. 6 and 7 delineate the impact of variable values of α_1 and α_2 on the classification accuracy within the VPN-nonVPN and Self-Collected datasets, respectively. It can be observed that within the respective ranges of α_1 and α_2, the classification accuracy generally shows an initial rise and then a decline. Analysis of the top three descending order accuracies in the VPN-nonVPN dataset identifies the corresponding α_1 and α_2 values as {0.5, 1.0, 2.0} and {2.0, 3.0, 1.0}, while those in the Self-Collected dataset are {0.25, 0.5, 0.1} and {2.0, 3.0, 4.0}, respectively. By considering the accuracy changes in both datasets, we assert that the optimal value of α_1 and α_2 are 0.5

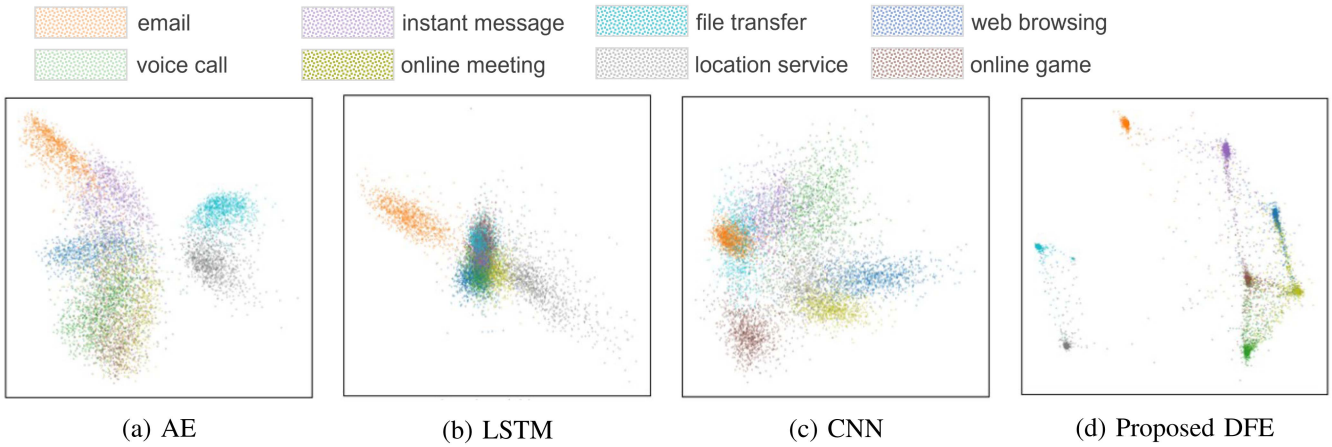


Fig. 5. Clusters of two-dimensional mapped neural activity that extracted by principal component analysis (PCA), with dots of different colors representing samples of different types of network traffics.

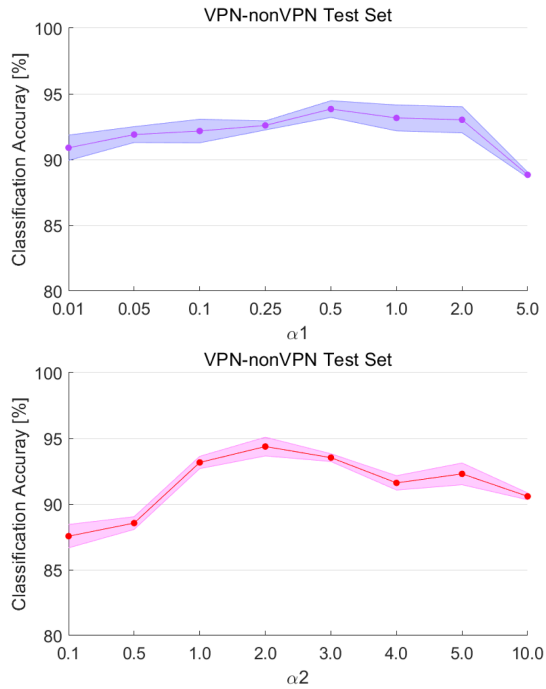


Fig. 6. Classification accuracy of different values of α_1 (top) and α_2 (bottom) on VPN-nonVPN test set.

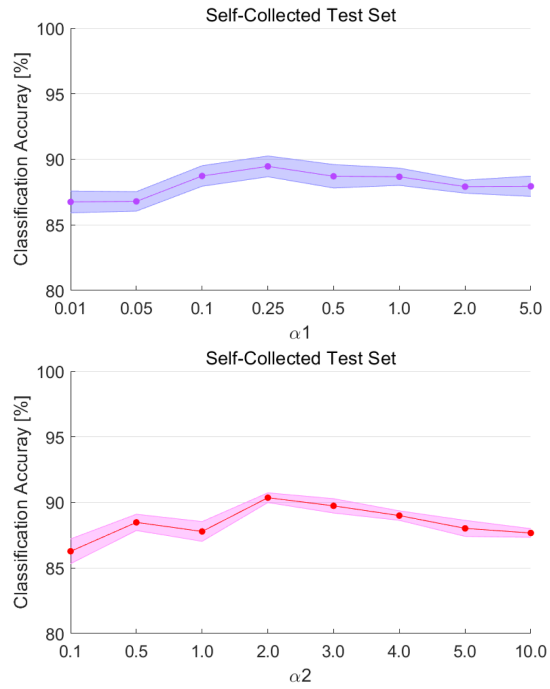


Fig. 7. Classification accuracy of different values of α_1 (top) and α_2 (bottom) on Self-Collecteed test set.

and 2.0, respectively, which coincide with our default parameter settings.

Fig. 8 depicts the effect of varying the number of template samples per category on the classification accuracy within the template library. We conducted 30 random tests to identify the impact, as revealed in the figure, of setting the number of template samples to the minimum value of 1, which not only results in the lowest classification accuracy but also exhibits the most significant fluctuations. This is attributed to the unique limitation of a single template that cannot cover all pattern representations of a category, ultimately causing the reduced accuracy during template matching. Furthermore, since the representation is versatile across the different randomly selected

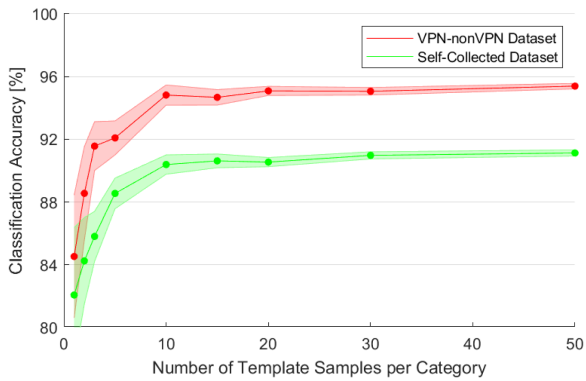


Fig. 8. The influence of the number of template samples per category.

samples, the fluctuations in accuracy are prominent. With the increase in the number of templates, the classification accuracy rises remarkably, and the fluctuations begin to diminish. Upon reaching approximately 20 templates per category, the classification accuracy of NTC attains its peak with minimal fluctuations. Considering the significant computational overhead associated with additional templates, we opt for 20 as the optimal number of templates per category.

VI. Conclusion

In this paper, we proposed a novel deep learning (DL) framework for network traffic classification (NTC), termed deep flow embedding (DFE). Unlike traditional artificial neural network, or machine learning methods that directly output classification result for the input of traffic data, the proposed DFE encodes the input to an embedding vector of numeric value. In addition, a feature compressor module was designed in the framework, which serves to compress the layer-by-layer feature information transmission in the DL backbone, while preserving computational accuracy. This enables the backbone to effectively eliminate noise, redundancy, as well as other irrelevant information from input data, facilitating more robust extraction of traffic flow features. Then, the distance between the output vector and template library is calculated and compared to identify the class of input traffic. The advantages of DFE can be concluded as: 1) For the problem of new traffic class or existed traffic with transmission protocol changes, DFE can still handle with high efficiency. It only needs to construct template vectors for these network traffic data, while does not require to collect a large amount of data to retrain the deep learning model. 2) DFE uses deep learning to build the backbone of feature extraction, which introduces the characteristics such as high computational accuracy, as well as high anti-noise robustness of deep learning, so as to achieve the goal of high-accuracy NTC. Through comparative experiments with other commonly-used algorithms, it is confirmed that our proposed DFE method has better performance. Future work should pivot towards the following objectives to further elevate the efficacy and accuracy of our framework:

1) Transition from manual feature selection practices to an automated, end-to-end feature extraction model that capitalizes on the latest advances in machine learning.
2) Refine the deep learning backbone, including its structure and the metric learning training methodologies, to optimize data processing bandwidth and elevate computational performance.
3) Investigate the integration of additional datasets encompassing a broader spectrum of network conditions and traffic types to enhance the framework's adaptability and scalability.
4) Examine the potential for implementing real-time adaptive mechanisms in the DFE that can fluidly adjust to evolving network traffic dynamics without human intervention.
5) Explore the possibility of hybrid model structures that combine the benefits of DFE with other promising machine learning paradigms to tackle the most challenging NTC scenarios.

Through these enhancements, the deep flow embedding framework may achieve not only a more profound level of traffic insight but also pave the way for autonomous network management solutions that can keep pace with the ever-changing digital landscape.

References

[1] Z. Xu and A. Zhang, "Network traffic type-based quality of experience (QOE) assessment for universal services," *Appl. Sci.*, vol. 9, no. 19, 2019, Art. no. 4107.

[2] J. Dai, X. Xu, H. Gao, and F. Xiao, "CMFTC: Cross modality fusion efficient multitask encrypt traffic classification in IIoT environment," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3989–4009, Nov.–Dec. 2023.

[3] M. Saqib, H. Elbiaze, and R. Glitho, "Adaptive in-network traffic classifier: Bridging the gap for improved QoS by minimizing misclassification," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 677–689, 2024.

[4] B. Ng, M. Hayes, and W. K. Seah, "Developing a traffic classification platform for enterprise networks with SDN: Experiences and lessons learned," in *Proc. 2015 IFIP Netw. Conf.*, 2015, pp. 1–9.

[5] P. Wang, S.-C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in *Proc. 2016 IEEE Int. Conf. Serv. Comput.*, 2016, pp. 760–765.

[6] Q. Wu, Q. Liu, Z. Jia, N. Xin, and T. Chen, "P4SQA: A p4 switch-based QoS assurance mechanism for SDN," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 4, pp. 4875–4886, Dec. 2023.

[7] B. Wang, Y. Su, M. Zhang, and J. Nie, "A deep hierarchical network for packet-level malicious traffic detection," *IEEE Access*, vol. 8, pp. 201728–201740, 2020.

[8] S. H. Almotiri, "Integrated fuzzy based computational mechanism for the selection of effective malicious traffic detection approach," *IEEE Access*, vol. 9, pp. 10751–10764, 2021.

[9] Y. Guo and D. Wang, "Feat: A federated approach for privacy-preserving network traffic classification in heterogeneous environments," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1274–1285, Jan. 2023.

[10] C. Sheng, Y. Yao, W. Li, W. Yang, and Y. Liu, "Unknown attack traffic classification in scada network using heuristic clustering technique," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2625–2638, Sep. 2023.

[11] S.-H. Yoon, J.-W. Park, J.-S. Park, Y.-S. Oh, and M.-S. Kim, "Internet application traffic classification using fixed ip-port," *Asia-Pacific Netw. Operations Manage. Symp.*, vol. 12, pp. 21–30, 2009.

[12] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "Portload: Taking the best of two worlds in traffic classification," in *Proc. 2010 INFOCOM IEEE Conf. Comput. Commun. Workshops*, 2010, pp. 1–5.

[13] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, "Unsupervised traffic classification using flow statistical properties and ip packet payload," *J. Comput. Syst. Sci.*, vol. 79, no. 5, pp. 573–585, 2013.

[14] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1135–1156, Second Quarter 2014.

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

[16] X. Yan, B. Liang, T. Ban, S. Guo, and L. Wang, "Traffics: A behavior-based network traffic classification benchmark system with traffic sampling functionality," in *Proc. Int. Conf. Neural Inf. Process.*, 2012, pp. 100–107.

[17] B. Qu, Z. Zhang, X. Zhu, and D. Meng, "An empirical study of morphing on behavior-based network traffic classification," *Secur. Commun. Netw.*, vol. 8, no. 1, pp. 69–79, 2013.

[18] M. Xu, W. Zhu, J. Xu, and N. Zheng, "Towards selecting optimal features for flow statistical based network traffic classification," in *Proc. 17th Asia-Pacific Netw. Operations Manage. Symp.*, 2015, pp. 479–482.

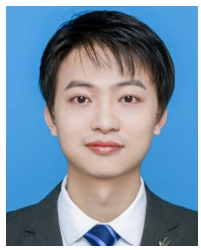
[19] M. Bhatia, V. Sharma, P. Singh, and M. Masud, "Multi-level P2P traffic classification using heuristic and statistical-based techniques: A hybrid approach," *Symmetry*, vol. 12, no. 12, 2020, Art. no. 2117.

[20] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Netw.*, vol. 26, no. 1, pp. 35–40, Jan./Feb. 2012.

[21] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988–2014, 2nd Quarter 2019.
[22] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54024–54033, 2019.
[23] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1962–1976, Jun. 2021.
[24] Y. Wang, X. Yun, Y. Zhang, C. Zhao, and X. Liu, "A multi-scale feature attention approach to network traffic classification and its model explanation," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 875–889, Jun. 2022.
[25] A. Telikani, A. H. Gandomi, K.-K. R. Choo, and J. Shen, "A cost-sensitive deep learning-based approach for network traffic classification," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 661–670, Mar. 2022.
[26] M. Lotfollahi, M. Jafari Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, 2020.
[27] K. C. Chiu, C. C. Liu, and L. D. Chou, "CAPC: Packet-based network service classifier with convolutional autoencoder," *IEEE Access*, vol. 8, pp. 218081–218094, 2020.
[28] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1171–1179.
[29] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Comput. Netw.*, vol. 165, 2019, Art. no. 106944. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619304669
[30] F. A. Md Zaki and T. S. Chin, "FWFS: Selecting robust features towards reliable and stable traffic classifier in SDN," *IEEE Access*, vol. 7, pp. 166011–166020, 2019.
[31] M. S. Sheikh and Y. Peng, "Procedures, criteria, and machine learning techniques for network traffic classification: A survey," *IEEE Access*, vol. 10, pp. 61135–61158, 2022.
[32] W. Li, X.-Y. Zhang, H. Bao, H. Shi, and Q. Wang, "Prograph: Robust network traffic identification with graph propagation," *IEEE/ACM Trans. Netw.*, vol. 31, no. 3, pp. 1385–1399, Jun. 2023.
[33] Y. Chen, T. Zang, Y. Zhang, Y. Zhou, L. Ouyang, and P. Yang, "Incremental learning for mobile encrypted traffic classification," in *Proc. ICC 2021 - IEEE Int. Conf. Commun.*, 2021, pp. 1–6.
[34] Y. Chen, Y. Tong, G. B. Hwee, Q. Cao, S. G. Razul, and Z. Lin, "Encrypted mobile traffic classification with a few-shot incremental learning approach," in *Proc. IEEE 18th Conf. Ind. Electron. Appl.*, 2023, pp. 40–45.
[35] G. Bovenzi et al., "Benchmarking class incremental learning in deep learning traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 1, pp. 51–69, Feb. 2024.
[36] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.
[37] X. Zhao, H. Qi, R. Luo, and L. Davis, "A weakly supervised adaptive triplet loss for deep metric learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop*, 2019, pp. 3177–3180.
[38] IANA, "Internet assigned numbers authority," 2024. [Online]. Available: http://www.iana.org/assignments/port-numbers
[39] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement*, C. Dovrolis, Ed. Berlin, Germany: Springer 2005, pp. 41–54.
[40] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 339–350, Aug. 2006.
[41] R. Antonello et al., "Deep packet inspection tools and techniques in commodity platforms: Challenges and trends," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 1863–1878, 2012.
[42] C. Liu and J. Wu, "Fast deep packet inspection with a dual finite automata," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 310–321, Feb. 2013.
[43] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "nDPI: Open-source high-speed deep packet inspection," in *Proc. Int. Wireless Commun. Mobile Comput. Conf.*, 2014, pp. 617–622.
[44] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep packet inspection over encrypted traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 213–226, Aug. 2015.
[45] N. Hua, H. Song, and T. V. Lakshman, "Variable-stride multi-pattern matching for scalable deep packet inspection," in *Proc. IEEE INFOCOM*, 2009, pp. 415–423.
[46] X. Ren, H. Gu, and W. Wei, "Tree-RNN: Tree structural recurrent neural network for network traffic classification," *Expert Syst. Appl.*, vol. 167, no. Aug. 2020, Art. no. 114363.
[47] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015.
[48] X. Peng and L. Sen, "Internet traffic classification using C4.5 decision tree," *J. Softw.*, vol. 20, no. 10, pp. 2692–2704, 2009.
[49] L. Hu and L. Zhang, "Real-time internet traffic identification based on decision tree," in *Proc. World Autom. Congr.*, 2012, pp. 1–3.
[50] A. Munther, A. Alalousi, S. Nizam, R. R. Othman, and M. Anbar, "Network traffic classification — A comparative study of two common decision tree methods: C4.5 and random forest," in *Proc. 2nd Int. Conf. Electron. Des.*, 2014, pp. 210–214.
[51] C. Wang, T. Xu, and X. Qin, "Network traffic classification with improved random forest," in *Proc. 11th Int. Conf. Comput. Intell. Secur.*, 2015, pp. 78–81.
[52] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for TCP traffic classification," *Comput. Netw.*, vol. 53, no. 14, pp. 2476–2490, 2009.
[53] G. Sun, T. Chen, Y. Su, and C. Li, "Internet traffic classification based on incremental support vector machines," *Mobile Netw. Appl.*, vol. 23, no. 4, pp. 789–796, 2018.
[54] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive Bayes predictions," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 5–15, Jan. 2013.
[55] G. Zhou Lin, Y. Xin, X. Xin Niu, and H. bai Jiang, "Network traffic classification based on semi-supervised clustering," *J. China Universities Posts Telecommun.*, vol. 17, pp. 84–88, 2010.
[56] Y. Zhao, J. Chen, G. You, and J. Teng, "Network traffic classification model based on MDL criterion," in *Advanced Multimedia and Ubiquitous Engineering*. Singapore: Springer, 2016, pp. 1–8.
[57] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, May 2019.
[58] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.
[59] M. Shen et al., "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 791–824, Firstquarter 2023.
[60] B. Sun, W. Yang, M. Yan, D. Wu, Y. Zhu, and Z. Bai, "An encrypted traffic classification method combining graph convolutional network and autoencoder," in *Proc. IEEE 39th Int. Perform. Comput. Commun. Conf.*, 2020, pp. 1–8.
[61] O. Aouedi, K. Piamrat, and D. Bagadthey, "A semi-supervised stacked autoencoder approach for network traffic classification," in *Proc. IEEE 28th Int. Conf. Netw. Protoc.*, 2020, pp. 1–6.
[62] R. Zhao et al., "A novel self-supervised framework based on masked autoencoder for traffic classification," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 2012–2025, Jun. 2024.
[63] Y. He and W. Li, "Image-based encrypted traffic classification with convolution neural networks," in *Proc. IEEE 5th Int. Conf. Data Sci. Cyberspace*, 2020, pp. 271–278.
[64] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z. H. Ling, "Encrypted network traffic classification using deep and parallel network-in-network models," *IEEE Access*, vol. 8, pp. 132950–132959, 2020.
[65] W. Chen, F. Lyu, F. Wu, P. Yang, G. Xue, and M. Li, "Sequential message characterization for early classification of encrypted internet traffic," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3746–3760, Apr. 2021.
[66] Y. Pan, X. Zhang, H. Jiang, and C. Li, "A network traffic classification method based on graph convolution and LSTM," *IEEE Access*, vol. 9, pp. 158261–158272, 2021.
[67] M. Balanici and S. Pachnicke, "Classification and forecasting of real-time server traffic flows employing long short-term memory for hybrid e/o data center networks," *J. Opt. Commun. Netw.*, vol. 13, no. 5, pp. 85–93, 2021.
[68] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 241–252, Feb. 2022.

[69] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Appl. Sci.*, vol. 9, 2019, Art. no. 3414.
[70] Z. Zou, J. Ge, H. Zheng, Y. Wu, C. Han, and Z. Yao, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *Proc. - 20th Int. Conf. High Perform. Comput. Commun., 16th Int. Conf. Smart City 4th Int. Conf. Data Sci. Syst., HPCC/SmartCity/DSS*, 2019, pp. 329–334.
[71] R. Sathyaa, J. Agrawalb, D. Roycand, and R. Dutta, "A self-adaptive and self-learning methodology for wireless intrusion detection using deep neural network," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 6, pp. 2084–2094, 2021.
[72] O. Aouedi, K. Piamrat, and B. Parrein, "Ensemble-based deep learning model for network traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4124–4135, Dec. 2022.
[73] Y. Hu, G. Cheng, W. Chen, and B. Jiang, "Attribute-based zero-shot learning for encrypted traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4583–4599, 2022.
[74] X. Xiao, W. Xiao, R. Li, X. Luo, H. Zheng, and S. Xia, "EBSNN: Extended byte segment neural network for network traffic classification," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 3521–3538, Sep./Oct. 2022.
[75] T.-L. Huoh, Y. Luo, P. Li, and T. Zhang, "Flow-based encrypted network traffic classification with graph neural networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1224–1237, Jun. 2023.
[76] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.
[77] Y. Zeng, Z. Qi, W. Chen, and Y. Huang, "Test: An end-to-end network traffic classification system with spatio-temporal features extraction," in *Proc. IEEE Int. Conf. Smart Cloud*, 2019, pp. 131–136.
[78] X. Wang, S. Chen, and J. Su, "APP-Net: A hybrid neural network for encrypted mobile traffic classification," in *Proc. IEEE INFOCOM 2020 - IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 424–429.
[79] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306–315, 2020.
[80] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *J. Netw. Comput. Appl.*, vol. 183–184, 2021, Art. no. 102985.
[81] S. Rezaei and X. Liu, "Multitask learning for network traffic classification," in *Proc. 29th Int. Conf. Comput. Commun. Netw.*, 2020, pp. 1–9.
[82] J. Cheng et al., "MATEC: A lightweight neural network for online encrypted traffic classification," *Comput. Netw.*, vol. 199, 2021, Art. no. 108472.
[83] A. Kolchinsky and B. D. Tracey, "Estimating mixture entropy with pairwise distances," *Entropy*, vol. 19, no. 7, 2017, Art. no. 361.
[84] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear information bottleneck," *Entropy*, vol. 21, no. 12, 2019, Art. no. 1181.
[85] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.
[86] L. Wang, H. Mei, and V. S. Sheng, "Multilevel identification and classification analysis of tor on mobile and PC platforms," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 1079–1088, Feb. 2021.
[87] M. Chen, M. He, L. Jin, K. Javeed, and X. Wang, "A network traffic classification model based on metric learning," *Comput., Mater. Continua*, vol. 64, no. 2, pp. 941–959, 2020.
[88] J. Cui, J. Long, E. Min, and Y. Mao, "WEDL-NIDS: Improving network intrusion detection using word embedding-based deep learning method," in *Modeling Decisions for Artificial Intelligence*. Cham, Switzerland: Springer, 2018, pp. 283–295.
[89] S. Zhao, Y. Zhang, and Y. Sang, "Towards unknown traffic identification via embeddings and deep autoencoders," in *Proc. Int. Conf. Telecommun.*, 2019, pp. 85–89.
[90] H. Gwon, C. Lee, R. Keum, and H. Choi, "Network intrusion detection based on LSTM and feature embedding," 2019, *arXiv:1911.11552*.
[91] V. Handika, J. E. Istiyanto, A. Ashari, S. R. Purnama, S. Rochman, and A. Dharmawan, "Feature representation for network intrusion detection system trough embedding neural network," in *Proc. Int. Conf. Comput. Eng., Netw., Intell. Multimedia*, 2022, pp. 349–352.

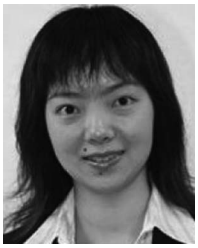
Zhijiong Wang received the bachelor's degree in electrical and information science and technology from Jilin University, Changchun, China, in 2014, and the master's degree in electrical and computer engineering in 2017 from University of Macau, Macau, China, where he is currently working toward the Ph.D. degree in electrical and computer engineering. His research interest includes biomedical embedded systems, digital VLSI design, machine learning, and artificial neural networks.



Anguo Zhang (Senior Member, IEEE) received the B.Sc. and M.S. degree from the School of Automation, Chongqing University, Chongqing, China, in 2012 and 2016, respectively, and the Ph.D. degree from the College of Physics and Information Engineering, Fuzhou University, Fujian, China, in 2022. From 2016 to 2022, he was a Senior Engineer with the Research Institute, Ruijie Networks Company Ltd. He is currently a Research Fellow with the Institute of Microelectronics, University of Macau, Macau, China. His research interests include brain-inspired computational intelligence, artificial neural networks, and computer vision.

Hung Chun Li received the Graduation degree from the National Taiwan University, Taipei, Taiwan, in 1990, majoring in electrical engineering, and the master's degree from the University of Florida, Gainesville, FL, USA. He has more than 28 years of experience in the semiconductor and IC design industry and is familiar with the global semiconductor supply chain ecosystem. In 1996, He was with ASIC Design Service Department, TSMC, where he was an Incharge of Developing Reference Flow and was an Initial Member with Open Innovation Platform (OIP). He was also a Technical Leader with Backend Design Engineering, Broadcom, and accomplished the first 10 GB ethernet switch chip tape out in mass production. Starting in 2002, he was with Cadence on physical implementation software which is used on hyperscale and nanometer technology design. He also led the R&D Team developing the STA timing and signal integrity analysis tools. After 2007, he joined GUC leading the Design Methodology Division which is specialized in advanced technology node and low power design. The team has accomplished several chip tapeouts in High Performance Computing (HPC) application using TSMC's 40 nm/28 nm/16 nm/10 nm/7nm technology. GUC is the design service company whose largest shareholder is TSMC. Before joining Foxconn, he was the Technical Director of Intel Foundry Service BU in 2015, responsible for design service program in China and Asia Pacific area. In 2018, he joined Foxconn Technology Group, as the Manager with Design Service Team under Foxconn ISSBG and acting CEO of Socle, which the design service company owned by Foxconn. In 2020, he established Zhuhai Lingyange Semiconductor Company Ltd., acting as CEO and CTO. The company provides advanced chips design service, IP design service, EDA reference flow and verification, production service, and system module design as well as IC channel sales.

Yadong Yin (Member, IEEE) received the B.S. degree in electrical engineering from Beijing Jiaotong University, Beijing, China, in 2003, and the Ph.D. degree in microelectronics from the Institute of Microelectronics, Chinese Academy of Sciences, Beijing, in 2009. From 2009 to 2014, he was with the Institute of Microelectronics, Chinese Academy of Sciences, Beijing, China. In 2015, he joined the School of Physics and Information Engineering, Fuzhou University, Fuzhou, China, where he is currently an Associate Research Fellow. His research interests include CMOS RF/mixed-signal integrated circuit, wireless power transfer systems, and ultralow power wireless communication systems.



Wei Chen (Senior Member, IEEE) received the Ph.D. degree in electrical and electronics engineering from the Department of Electrical and Electronics Engineering, University of Melbourne, Melbourne, VIC, Australia, in 2007. She is currently a Full Professor and the Director with the Center for Intelligent Medical Electronics, School of Information Science and Technology, Fudan University, Shanghai, China. Her research interests include health monitoring, sleep monitoring and medical electronics.

Chan Tong Lam (Senior Member, IEEE) received the B.Sc. (Eng.) and M.Sc. (Eng.) degrees from Queens University, Kingston, ON, Canada, in 1998 and 2000, respectively, and the Ph.D. degree from Carleton University, Ottawa, ON, Canada, in 2007. From November 2000 to May 2003, he was with Sigpro Wireless, Inc., Ottawa, as a 3rd Generation (3G) Systems Engineer. From January 2004 to August 2007, he participated with European Wireless World Initiative New Radio Project (within the work package of physical layer design). From November 2007 to August 2008, he was with the Bureau of Telecommunications of Macao, as a Technical Officer. Since September 2008, he has been an Associate Professor of computing program with Macao Polytechnic Institute, Macao. Since June 2016, he has also been the Coordinator of the Computing Program. His research interests include mobile communications for 5th Generation System, communication technology for Internet of Things, computer vision for smart city, and mobile learning for computer programming.

Peng Un Mak (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, and the M.Sc. and Ph.D. degrees in electrical engineering from Michigan State University, East Lansing, MI, USA. Since 1997, he has been an Assistant Professor with the Department of Electrical and Computer Engineering, Faculty of Science and Technology, University of Macau, Macau, China. From 2012 to 2013, he was a Visiting Scholar with the Department of Physiology and Biophysics, School of Medicine, Anschutz Medical Campus, University of Colorado, Denver, CO, USA. In 2018, he was also a Visiting Fellow of Clare Hall, University of Cambridge, Cambridge, U.K. His research interests include biosignal extraction and processing, bioelectromagnetism, human body communication, and body sensor networks. He is also a Life Member with Phi Kappa Phi and Clare Hall. He is an Invited Member of the Eta Kappa Nu (currently IEEE-HKN).

Mang I Vai (Senior Member, IEEE) received the Ph.D. degree in electrical and electronics engineering from the University of Macau, Macau, in 2002. Since 1984, he has been performing research in the areas of digital signal processing and embedded systems. He is currently a Coordinator with the State Key Laboratory of Analog and Mixed-Signal VLSI and an Associate Professor of electrical and computer engineering with the Faculty of Science and Technology, University of Macau.

Yueming Gao (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Fuzhou University, Fuzhou, China, in 2010. Since 2004, he has been involved in research in the areas of bioelectromagnetism and biomedical circuits and systems. He is currently a Professor with the School of Physical and Information Engineering, Fuzhou University.

Sio Hang Pun (Senior Member, IEEE) received the master's degree in computer and electrical engineering from the University of Porto, Porto, Portugal, in 1999, and the Ph.D. degree in electrical and electronics engineering from the University of Macau, Macau, in 2012. Since 2012, he has been an Associate Professor with the State Key Laboratory of Analog and Mixed-Signal VLSI, University of Macau, Macau, China. His research interests include biomedical electronic circuits, miniaturized sensors for biomedical applications, and human body communication.