Key Technologies for AI-Driven Network Traffic Classification Workflow and Data Distribution Shift

Zhao Jianchao[1], Geng Zhaosen[1], Li Zeyi[2], Wang Pan[3]

(1. Cable Products Business Department, ZTE Corporation, Shenzhen 518057, China;
2. School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
3. School of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

DOI: 10.12142/ZTECOM.202601006

https://kns.cnki.net/kcms/detail/34.1294.TN.20260112.1759.002.html, published online January 13, 2026

Manuscript received: 2025-01-11

Abstract: With the evolution of next-generation network technologies, the complexity of network management has significantly increased, and the means of network attacks are diversified, bringing new challenges to network traffic classification. This paper presents a general AI-driven network traffic classification workflow and elaborates on a traffic data and feature engineering framework. Most importantly, it analyzes the concept and causes of data distribution shifts in network traffic, proposing detection methods and countermeasures. Experimental results on real traffic collected at different time intervals show that application evolution can induce data distribution shifts, which in turn lead to a noticeable degradation in traffic classification performance. Comparative drift detection experiments further confirm that such shifts are more evident over long-term intervals, while short-term traffic remains relatively stable. These findings demonstrate the necessity of incorporating drift-aware mechanisms into AI-driven network traffic classification systems.

Keywords: traffic classification; traffic identification; deep learning; data distribution shift; concept shifting

Citation (Format 1): Zhao J C, Geng Z S, Li Z Y, et al. Key technologies for AI-driven network traffic classification workflow and data distribution shift [J]. ZTE Communications, 2026, 24(1): 34 - 44. DOI: 10.12142/ZTECOM.202601006

Citation (Format 2): J. C. Zhao, Z. S. Geng, Z. Y. Li, et al., "Key technologies for AI-driven network traffic classification workflow and data distribution shift," *ZTE Communications*, vol. 24, no. 1, pp. 34 - 44, Mar. 2026. doi: 10.12142/ZTECOM.202601006.

1 Introduction

Network traffic classification (TC), as an essential means for network management and security, has received significant attention from academia and industry since the late 1990s. It has been well applied in quality of service/quality of experience (QoS/QoE) management, network resource optimization, congestion control, intrusion detection, etc. [1] With the rapid development of new-generation network technologies (B5G/6G, Internet of Things, celestial and terrestrial integrated networks, etc.), network technology is moving towards high autonomy of self-healing, self-management, self-optimization and self-protection, and the network traffic classification technology plays a key role as one of the decision-making tools for the network service and security management[2]. However, with the ubiquitous access of a large number of heterogeneous terminals, the network shows a high degree of dynamism, heterogeneity and complexity. This brings new challenges to network traffic classification technology. In particular, the frequent upgrading of legacy applications, the continuous emergence of new applications, and the gradual downgrading of silent applications have left network TC technology "one step behind", unable to keep pace with dynamic application change.

The development of TC technology has roughly gone through three stages. The first phase is based on ports or deep packet inspection (DPI) to achieve TC. However, as applications increasingly adopt technologies such as tunnelling, encryption, and random ports, coupled with the security risk of user privacy leakage, these technologies quickly become ineffective. The second phase mainly uses machine learning (ML) based methods to learn the intrinsic laws of different business/application/attack traffic characteristics and implement the delineation of various applications in the data space to achieve traffic classification. However, such methods need to extract high-quality traffic features as the training basis of ML, and

This work was supported by ZTE Industry-University-Institute Cooperation Funds under Grant No. HC-CN-20220607009.

the extraction and selection of these features are highly dependent on the experience of network experts and time-consuming. With the rapid development of cloud computing, big data, especially deep learning (DL) and high-performance computing technologies, feature learning of massive traffic data has become possible, bringing new extension space to the TC field. DL has three excellent features: automatic feature extraction that can reveal more profound data laws, a large number of mature applications in vision/image/text/voice models, and the ability to address the gaps in ML-based TC methods. In recent years, DL-based TC techniques (hereafter referred to as DL-TC; AI-TC in subsequent text denotes ML/DL-TC) have been proposed and sparked new research enthusiasm. These include methods based on the convolutional neural network (CNN), the auto encoder (AE), the multilayer perceptron (MLP), long short-term memory (LSTM), and the generative adversarial network (GAN), which have achieved better classification performance than ML-TC[3-9].

Many research institutes and personnel are actively researching and developing efficient AI-TC algorithms and models, but there are still many problems for practical deployment and operation. Table 1 shows the comparison between academia and industry in AI-TC.

The data distribution shift has now become a research hotspot in AI, but most studies focus on classical AI fields such as images, vision, speech, and texts, and few studies have ventured into network traffic classification. This paper focuses on the data shift problem in AI-TC, describing its concepts, challenges, and critical techniques. The contributions of this paper are as follows:

1) A generic framework for AI-driven sustainable learning of network traffic classification systems is proposed;

2) The existing research progress on the data distribution shift problem is summarized, and a data distribution shift detection method for AI-TC is proposed;

3) The current advances in continuous learning research are summarized and a continuous learning method for AI-TC is presented;

4) The challenges and future research directions in data distribution shift detection and continuous learning within the context of AI-TC are identified.

2 A Generic Framework for Sustainable Learning of AI-TC System

2.1 AI-TC System Framework

To cope with the three major challenges of dynamic changes in the network environment, rapid evolution of business applications, and continuous upgrading of privacy protection in the new generation of communication networks, the AI-TC system has to be an iterative optimization process with continuous learning. The generic end-to-end AI-TC workflow is illustrated in Fig. 1. From the perspective of an end-to-end machine learning lifecycle, a common framework for sustainable learning AI-TC classification systems includes the definition of AI-TC classification requirements/design principles, traffic data engineering (TDE), feature engineering and model development and evaluation, model interpretation, deployment, model monitoring, and continuous learning phases. The separate phases are detailed below:

• Design principle: It primarily aims to define detailed classification requirements and development principles for AI-TC. These requirements include classification granularity, application scenarios, data sources, and real-time/non-real-time considerations, while the development principles encompass reliability, robustness, security, adaptability, etc.

• TDE: It builds datasets for training and testing. This includes considerations for data sources (baseline data sources, real-time data sources, etc.), traffic collection, traffic sampling, preprocessing (background traffic, redundant data, etc.), and traffic labeling (including both manual and ML-based labeling). Furthermore, after model deployment, it is essential to continuously gather and sample new traffic sample representa-

Table 1. Comparison between academia and industry in AI-TC

	Academia	Industry
Classification requirements	Modeling on training datasets for optimal performance	Comprehensive study of training/classification costs, efficiency, continuous operation, credibility, etc.
End-to-end TC	Focus on the modeling process before the TC model is deployed	Increased focus on monitoring optimization of TC models after deployment
Training data	Static/obsolete, well labeled, noise known	Continuous/changing, no/wrong labeling, noise unknown
Costs	Mostly unconcerned	Great concern
Data distribution shift	Mostly unconcerned	Great concern
Continuous learning	Mostly unconcerned	Great concern
Interpretability	Mostly unconcerned	Consideration
Computational complexity	Focus more on training fast	More concerned with reasoning fast

TC: traffic classification

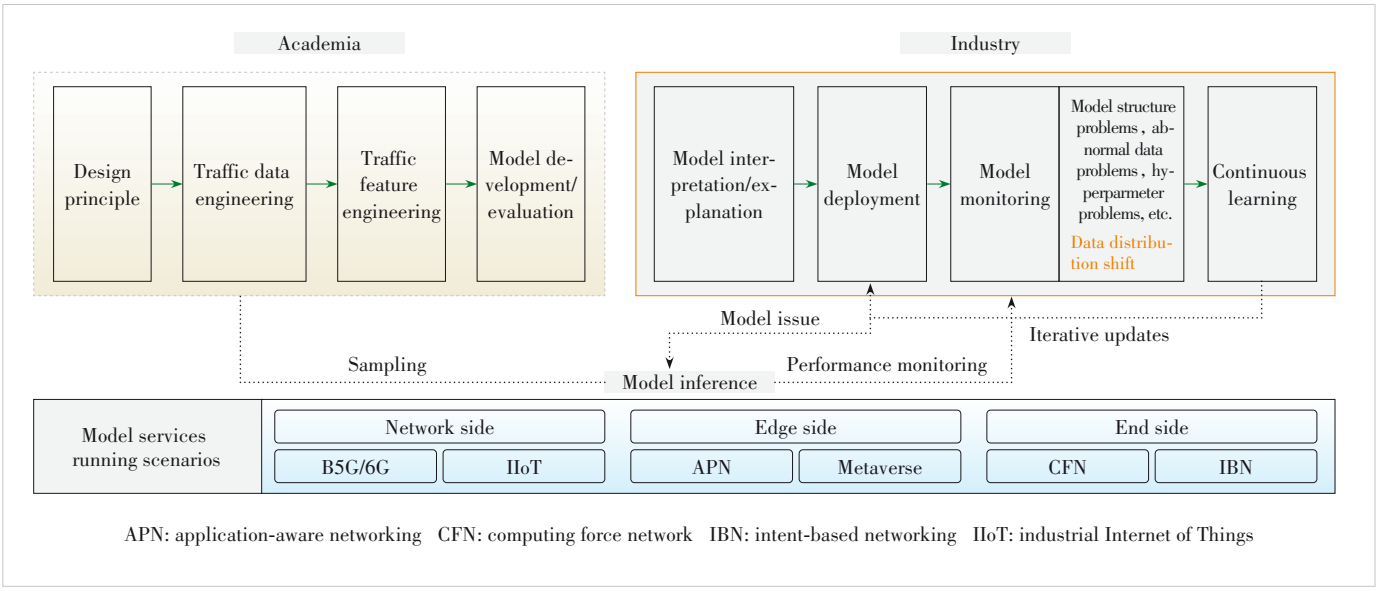


Figure 1. Generic end-to-end AI-TC workflow

tives of typical scenarios (e.g., through active learning) to support ongoing learning.

• Traffic feature engineering (TFE): It performs feature extraction, selection, representation, and reduction for network traffic data to construct an optimal feature subset in accordance with the design principles of the AI-TC classification system.

• Model development/evaluation: This session involves the choice of learning methods (supervised, semi-supervised, unsupervised, and weakly supervised), training methods (centralized/distributed training, federated learning, etc.), whether to pre-train or not, and whether to use the classical model for transfer learning or not. In addition, based on the TC design principles, the corresponding performance evaluation metrics are defined, including accuracy, precision, recall, F1 score, AUC, etc. In addition, the computational complexity, time complexity, computational resources (CPU/memory/flash memory) and time (training/inference) required for training/reasoning of the model are also considered.

• Model interpretation/explanation (XAI): In a narrow sense, model interpretability mainly solves the trustworthiness of classification model users (e.g., operators). Specifically, it focuses on solving the "black box" problem of AI-TC models so that users of the classification model (e.g., operators) can trust the model and have confidence in using the model. In a broader sense, model interpretability also includes transparency and fairness. The former primarily aims to provide model transparency for developers, turning the "black box" into a "white box", which is convenient for model problem diagnosis and iterative optimization. The latter is to detect whether the model is biased, e.g., whether the classification model favors high-value users while neglecting low-value users, especially in the application scenario of bandwidth guarantee based on refined application classification.

• Model deployment: According to different application scenarios, the reasoning environment of classification models can be divided into the network side, the edge side and the terminal side. The model deployment includes model sending methods (pull/push/subscribe, etc.), update strategies, training participation modes (e.g., the federated learning mode), traffic sampling methods (e.g., active learning), and performance parameter reporting.

• Model monitoring: By comprehensively monitoring the status of the classification model and the real-time data flow, this link can detect key issues such as classification system failure and classification model degradation promptly, thereby triggering continuous learning and driving a new round of iterative updates. This is crucial to the robustness of the entire classification system. In addition to general hardware/software failures of the classification system, the scope of monitoring also includes specific machine learning failures such as hyperparameter problems and abnormal data problems. The most important is the data distribution shift, which is particularly severe in network traffic classification.

• Continuous learning: This session is triggered by model monitoring, which initiates a series of iterative optimization and continuous learning from TDE, TFE, as well as model development evaluation, deployment, etc., which is the key to keeping the whole classification system with high adaptability, robustness and reliability.

The following content focuses on an in-depth elaboration of two issues: data distribution shifts and continuous learning. Their concepts, existing technologies, and applications in the context of AI-TC are introduced, and open questions and future research directions are presented.

2.2 Flow Data and Feature Engineering

The traffic data distribution shift problem is inextricably linked to traffic data and feature engineering, and this paper proposes an AI-driven workflow for network traffic classification data and feature engineering, as shown in Fig. 2.

1) Data source: a combination of baseline data and real-time streaming data. Baseline data mainly consists of public and self-built private datasets. In addition, real-time streaming data from the real network being served is also collected to meet the need for continuous learning of new traffic data class distributions.

2) Traffic acquisition and preprocessing: Considering the huge amount of real-time streaming data, it is necessary to design a reasonable traffic sampling strategy to collect representative traffic samples. In addition, pre-processing, i. e., "traffic distilling", is also required because real-time flow data often contains background traffic, redundant groups and other noisy data.

3) Traffic labeling: The traffic is labeled with business types, applications, or attacks, i.e., the ground truth, which is an extremely critical step in dataset construction and will directly affect the performance of the AI-TC model. Developing the automated traffic annotation capability independent of network experts is the future development trend in traffic labelling[10].

4) Feature extraction, selection, and compression: It refers to extracting representative flow features from raw packets, whether using ML or DL. Flow features are mainly classified into raw packet bytes and flow attributes. The former is often used for packet-grained classification in real-time scenarios, while the latter is classified into packet-, flow-, and session-level features according to the granularity of the flow attributes, which are primarily in the form of spatial (packet length, number of packets, flow length, etc.), temporal (inter-packet time interval, flow duration, etc.) and statistical features (expectation, variance, etc.). A typical set of network traffic features is shown in Table 2. To reduce the complexity of the model, it is also necessary to select the most representative characteristics according to certain principles, which involves the feature selection method[11]. The features need to be compressed to further reduce the model parameters and the complexity of model training, usually using AE, principal component analysis (PCA), and other dimensionality reduction methods.

5) Feature representation: After extracting traffic features for model training, it is often necessary to choose a suitable way to express the traffic feature information. Common methods of expression include 2D vectors, images, byte sequences, graphs, etc. This forms a feature set for model training.

6) Data storage and retrieval: Traditional network traffic datasets are often stored as PCAP raw files or stream feature csv files. However, data formats, models, storage, and retrieval methods will face new challenges when building large-scale massive datasets.

3 Data Distribution Shift in AI-TC

3.1 Concept

After the AI-TC classification model is deployed to the real network environment, the data distribution of the application traffic becomes different from that during the training period, which is called the data distribution shift (DDS). One fundamental assumption of ML systems is that the training set and unseen data come from the same static distribution. In the model development and evaluation phase, the testing set represents the unseen data, and the model's performance on the testing set reflects the model's generalization ability. However, in practice, this is often not the case. The unseen data is prone to change (e.g., application updates, emergence of new applications, etc.), i.e., the DDS phenomenon occurs, leading to degraded model performance. The DDS phenomenon arises from two reasons: first, the training set is limited by data collection/sampling, annotation, and preprocessing methods, which are unable to represent real-world data distributions re-

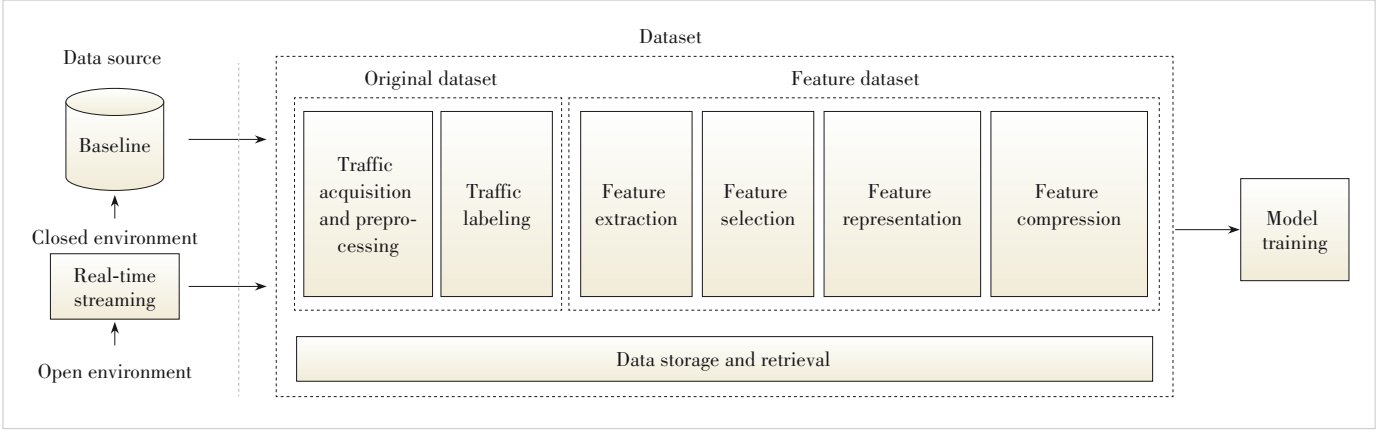


Figure 2. AI-driven network traffic data and feature engineering

Table 2. A typical collection of network traffic features (partial)

Flow Feature	Category	Description	Feature Calculation Method
Flow 5-tuple	Flow index	src/sp/dst/dp/protocol	Serialized regular preprocessing
TCP slide window	TCP window	TCP flow control parameters	Serialized regular preprocessing
TLS handshake packet information	TLS fingerprint	Handshake types, cipher suites, content types, key length, etc	Serialized regular preprocessing
Packet length sequence	Packet-related	A sequence of packet lengths in the stream. It may contain upstream, downstream, and bidirectional sequences as needed.	Packet length variance (max/min/ave/std)
Packet arrival times		A time sequence of packet arrivals in a diversion. Upstream, downstream and bidirectional sequences may be included as needed.	Packet time variance (max/min/ave/std)
Flow length related	Flow-related	Total number of flow bytes per unit of time, which may include upstream, downstream, and bidirectional as needed.	Multi-flow length variance (max/min/ave/std)
Flow duration		TCP flow duration UDP flow duration can be increased if NP resources are sufficient.	Multi-flow duration variance (max/min/ave/std)

NP: network processing TCP: transmission control protocol TLS: transport layer security UDP: user datagram protocol

alistically; second, real-world data is not static, but dynamically changes with new applications and application updates. The solution to the AI-TC data shift problem should achieve three objectives: 1) detecting the drift with the minimum number of traffic samples; 2) accurately characterizing the shift characteristics of the traffic data distribution, and preferably identifying shift samples from test data; 3) quantifying the degree of malignancy of the drift as much as possible.

3.2 Categories

DDS includes three subtypes: the covariate shift (CS), the concept drift (CD), and the label shift (LS). If the input of a classification model is defined as X, its probability distribution is $P(X)$; if the output is Y, its probability distribution is $P(Y)$. Under the supervised learning paradigm, training data can be viewed as a set of samples conforming to the joint probability distribution of $P(X, Y)$, and the ML/DL-based TC classification model aims to build the model of $P(Y|X)$.

CS refers to changes in the distribution of input traffic data $P(X)$ while keeping the conditional probability $P(Y|X)$ unchanged. There are three causes of this drift: 1) The bias in the traffic data collection/sampling process. For example, fewer samples are collected when constructing the traffic dataset due to less Distributed Denial of Service (DDoS) traffic. When DDoS attacks occur in the natural network environment, the traffic distribution differs significantly from that in training; 2) data augmentation operations performed to alleviate the class imbalance problem of the training dataset, which cause the input distribution to be inconsistent with reality, such as over-sampling, SMOTE, or GAN; 3) the model training process causing the traffic data distribution to change. In particular, active learning, which selects the most representative samples based on some assumptions or prior information, making the distribution of training input data differ from the real world.

CD, known as posterior drift, refers to a situation where the input distribution remains unchanged, but given an input, the conditional probability distribution of the output changes, that is, $P(Y|X)$ changes while $P(X)$ remains constant. In network traffic classification, it often refers to the emergence of new applications and iterative updates to old applications. This causes instances previously predicted as Application A to now be predicted as unknown or mistakenly classified as Application B, while classification models use the same input. The CD issue is a relatively concentrated problem in previous research on network traffic classification and intrusion detection involving the drift of concepts.

LS, also known as prior drift or target drift, refers to a situation where $P(Y)$ changes while $P(X|Y)$ remains constant. In other words, when the output distribution changes, the input distribution remains unchanged for a given output. Taking DDoS attack traffic as an example, AI-TC primarily identifies it as a DDoS attack based on the typical characteristic of constant inter-packet time intervals (statistically, this means slight variance). If AI-TC classifies traffic as a DDoS attack flow, the rule based on typical flow characteristics remains unchanged, which means $P(X|Y)$ remains constant. However, due to real network conditions, DDoS attack behavior suddenly surges, causing an increase in this type of attack traffic, which means that $P(Y)$ distribution changes (compared with the DDoS probability distribution in the training dataset). When the input traffic distribution changes, causing a CS phenomenon, the output distribution often changes accordingly, thus leading to the LS phenomenon.

3.3 Causes of Formation

There are many causes of distribution drifts in flow data, which are broadly classified into the following two categories.

3.3.1 Causes of CS/LS

Since CS and LS often occur concurrently, their underlying causes are fundamentally similar.

1) Flaws in data collection methods for the training dataset lead to CS or LS. Due to limitations in data collection, the distribution of some traffic categories in the training set differs from that in real networks. For instance, malicious attacks occur more frequently in real networks, but simulating attack traffic is challenging during data collection for training, resulting in a smaller amount of collected traffic.

2) Feature changes can lead to CS/LS. For example, introducing new features, such as the device type (Apple or Android), for fine-grained mobile application identification in the AI-TC model can result in CS/LS.

3) The use of data augmentation methods to address class imbalance may cause CS/LS.

4) CS/LS arises during the model learning process. For instance, selecting the most representative samples based on heuristic rules for active learning can result in CS/LS.

3.3.2 Causes of CD

CD is often the result of new applications emerging, old applications getting updated, and inactive applications being taken down. This is particularly significant in security scenarios like intrusion detection. For instance, in order to evade detection, hackers continuously enhance and modify their attack methods using various traffic obfuscation techniques.

4 Shift Detection

4.1 Traditional Approaches

Traditional methods infer whether the DDS problem occurs by monitoring the changes in metrics such as accuracy, precision, recall, F1-score, and AUC-ROC. However, these metrics need to be compared with the ground truth. The model is often unable to obtain the real value in real-time or after a considerable delay, so the method is impractical. Lipton et al.[12] proposed a black-box shift estimation method to detect and quantify the degree of shift in the data distribution without the need for actual data labels. This method designs a black-box predictor to reduce data dimensionality and, by utilizing the reversibility of a confusion matrix, predicts whether a data distribution shift has occurred and quantifies the degree of the shift.

4.2 Statistical Methods

The statistical method is to compare the statistical values of the source distribution and the target distribution, such as min, max, mean, median, variance, skewness, and kurtosis. For example, statistics on the expected value and variance of packet lengths in network flow features during model inference are compared with those of the training dataset to determine whether there is any shift. Google's TensorFlow extension plugin currently supports some of these features. However, the statistical values in the training set and those during the inference process may be quite similar, making it difficult to determine whether a data distribution shift has occurred. Currently, shift detection based on statistical methods can be roughly categorized into two types: univariate detection and multivariate detection.

4.3 Two-Sample Hypothesis Test

Another method is using a two-sample hypothesis test (referred to simply as the two-sample test). This test serves to ascertain whether a substantial statistical disparity exists between two datasets. Detecting such a difference indicates a reduced likelihood of random variations within the data. Furthermore, it suggests a greater probability that the two datasets stem from distinct distributions, signifying a shift in the data distribution. A typical two-sample test method is the Kolmogorov-Smirnov test, or the K-S test, which does not require prior knowledge of the data distribution, but detects the difference in the statistical level between the two datasets from the training set and the real-time flow to determine the shift. However, the disadvantage of this method is that it cannot work in a high-dimensional space. Thus, high-dimensional data must first be downscaled before applying the K-S test for statistical level difference analysis. Fig. 3 shows a framework for detecting shifts in network traffic distribution based on the two-sample test. The framework performs a dimensionality reduction operation on data from two distributions—the training

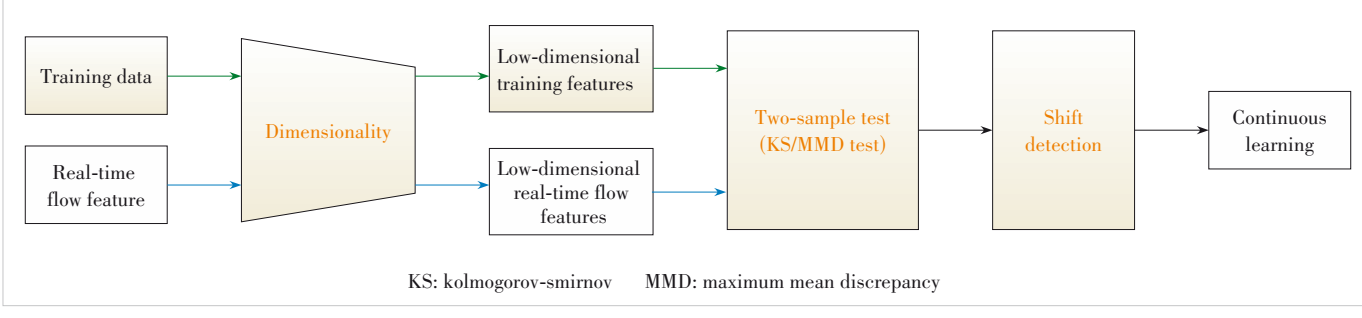


Figure 3. Network traffic distribution shift detection based on a two-sample test

set and the real-time flow—to form low-dimensional training features and real-time flow features, feeds them into the K-S two-sample test, integrates several statistics for shift detection, and finally performs quantitative analyses of the shift degree. These include various dimensionality reduction methods, PCA, AE, etc. There are also many methods for two-sample testing, such as maximum mean discrepancy (MMD)[13] and learned Kernel MMD[14].

Fig. 3 shows the whole shift detection process. First, the training feature set and the real-time flow feature are reduced to low-dimensional features, respectively with dimensionality reduction means like PCA. The low-dimensional training features are referred to $X = x_1, x_2, \cdots, x_n$, while the low-dimensional real-time flow features are $\hat{X} = \widehat{x_1}, \widehat{x_2}, \cdots, \widehat{x_n}$. Then the distance between the two distributions is calculated using MMD. The calculation formula is: $\mathrm{MMD}\left(X, \hat{X}\right) = \left| \frac{1}{n}\sum_{i=1}^{n}\phi\left(x_i\right) - \frac{1}{m}\sum_{j=1}^{m}\phi\left(\widehat{X}_j\right)\right|^2$, which calculates the squared Euclidean distance between the mean of samples mapped to the feature space in X and $\hat{X}$ separately. When the value of MMD is 0, we consider the two distributions to be the same. Thus, MMD measures the difference between the distributions of samples by comparing their means in feature space and thus can be used to detect shift between domains.

4.4 Two-Sample Testing Experiments

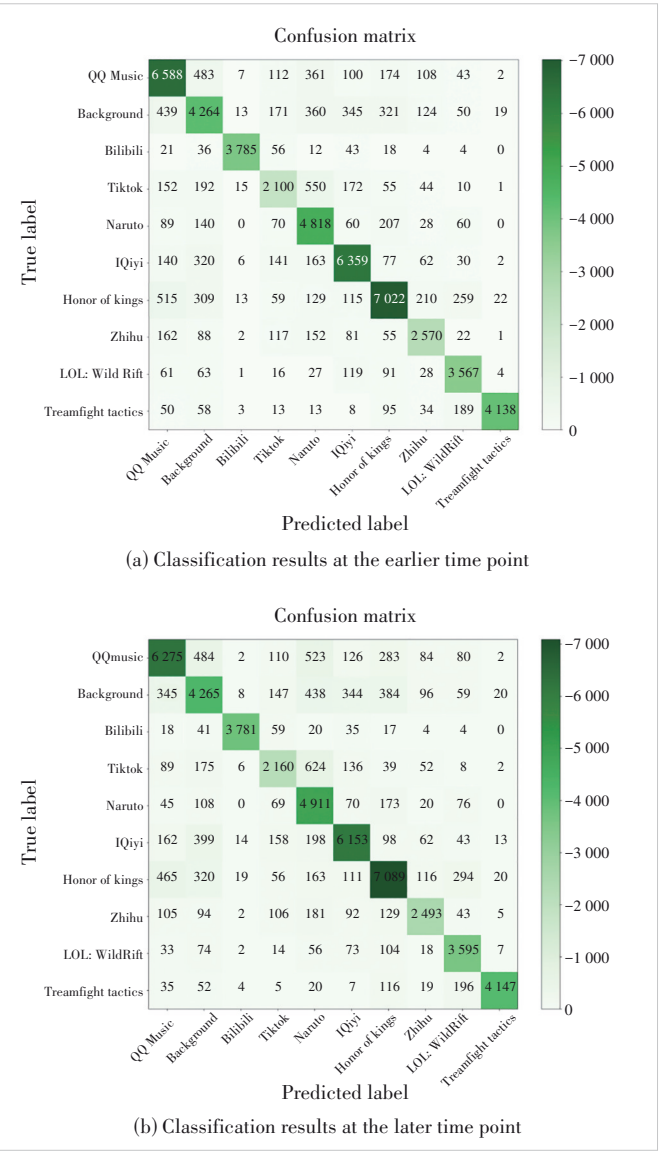
We utilized real traffic collected at different time intervals for the same application as our dataset. On the personal terminal, PCAPdroid is used to mark network traffic, and on the router, Wireshark is used to capture it. Finally, the two are compared and filtered to obtain the traffic data required by the experiment. CICFlowMeter is then used to calculate the network traffic characteristics for each application's PCAP file to get the corresponding csv file. The applications contained in the dataset and the corresponding number of streams are shown in Table 3.

Table 3. Dataset and the corresponding number of streams

App	Flow
QQmusic (Music)	39 465
LOL:Wild Rift (Game)	19 841
Naruto (Game)	27 240
Zhihu (Sociality)	16 643
Bilibili (Video)	20 014
Teamfight Tactics (Game)	23 718
IQiyi (Video)	36 740
Tiktok (Video)	16 640
Honor of Kings (Game)	42 734
Background (Log)	30 000

We employed a consistent traffic classification model, and the results are shown in Fig. 4 and Table 4. Traffic of the same application collected at two different time points with a one-month interval was used for evaluation. Fig. 4a presents the confusion matrix for traffic collected at the earlier time point, while Fig. 4b shows the results for traffic collected at the later time point. As can be observed, the classification performance of QQ Music exhibits a noticeable degradation across the two time points, with the number of correctly recognized samples decreasing from 6 588 to 6 275, whereas the performance of most other applications remains relatively stable. This observation suggests the presence of a potential data distribution shift, which motivates the subsequent drift detection experiments.

The experimental results show that the classification performance of QQ Music degrades noticeably, which is consistent



(a) Classification results at the earlier time point

(b) Classification results at the later time point

Figure 4. Confusion matrices of network traffic classification results at two different time points

Table 4. Classification results of network traffic at different times

	Traffic at Time Point T_1				Traffic at Time Point T_2			
	precision	recall	f1-score	support	precision	recall	f1-score	support
QQ Music	0.802	0.826	0.814	7 978	0.829	0.787	0.807	7 978
Background	0.716	0.698	0.707	6 106	0.709	0.698	0.704	6 106
Bilibili	0.984	0.951	0.968	3 979	0.985	0.950	0.967	3 979
Tiktok	0.736	0.638	0.683	3 291	0.749	0.656	0.700	3 291
Naruto	0.732	0.880	0.799	5 472	0.688	0.897	0.779	5 472
IQiyi	0.859	0.871	0.865	7 300	0.861	0.843	0.852	7 300
Honor of Kings	0.865	0.812	0.838	8 653	0.841	0.819	0.830	8 653
Zhihu	0.800	0.791	0.795	3 250	0.841	0.767	0.802	3 250
LOL: Wild Rift	0.842	0.897	0.869	3 977	0.817	0.904	0.859	3 977
Teamfight Tactics	0.988	0.899	0.942	4 601	0.984	0.901	0.941	4 601
Accuracy	0.828	0.828	0.828	0.828	0.822	0.822	0.822	0.822
Macro avg	0.832	0.826	0.828	54 607	0.830	0.822	0.824	54 607
Weighted avg	0.831	0.828	0.828	54 607	0.827	0.822	0.822	54 607

with the drift detection outcomes. To further investigate this phenomenon, paired-sample drift detection was performed by comparing traffic collected at two different time points. As illustrated in Fig. 4a, the traffic features from two closely spaced time points are highly similar and largely overlap, indicating no observable data drift. In contrast, Fig. 4b demonstrates a clear distribution shift when comparing traffic collected at two time points separated by a longer interval. These results suggest the presence of a data distribution shift, which motivates subsequent drift detection analysis.

The corresponding drift detection results are summarized in Table 5. For the one-day interval data, most methods report no drift, except for Spot-the-diff. For the one-month interval data, the majority of methods successfully detect drift, with the exception of the Least-Squares Density Difference method. In terms of efficiency, Kolmogorov-Smirnov, Cramérvon Mises, and the mixed-type tabular data methods exhibit the lowest execution times, whereas MMD and Spot-the-diff incur the highest computational cost.

The distance for detecting drift in Table 5 denotes a function or metric utilized in measuring the variation or resemblance of data or concepts. For instance, MMD is a representative technique for drift detection based on data distributions. The fundamental concept of MMD is that if the moments of any two data distributions are identical, then these two distributions are consistent. Conversely, the two distributions have

Table 5. Experimental results of data drift in response to different time periods

Data	Methods	Drift Occurs	Distance (unitless)	Execution Time/s
One-month interval	Kolmogorov-Smirnov	√	/	0.062
	Maximum mean discrepancy	√	0.192 245 841	1.385
	Chi-Squared	√	/	0.232
	Cramérvon Mises	√	/	0.030
	Least-squares density difference	√	0.239 282 097	0.398
	Spot-the-diff	×	0.551 293 545	1.293
	Mixed-type tabular data	√	/	0.069
One-day interval	Kolmogorov-Smirnov	×	/	0.064
	Maximum mean discrepancy	×	0.000 526 399	1.449
	Chi-Squared	×	/	0.145
	Cramérvon Mises	×	/	0.025
	Least-squares density difference	√	0.003 137 094	0.391
	Spot-the-diff	×	0.054 495 389	1.286
	Mixed-type tabular data	×	/	0.069

drifted if a particular moment exhibits divergence. If the MMD's value is 0, it demonstrates that the two distributions are identical and have not drifted. The approach employs a Gaussian kernel function to capture changes in the data's higher-order moments. A larger MMD value signifies a greater disparity between the two distributions, implying a higher degree of drift. However, the flow features do not provide sufficient information about QQ Music. For instance, the Spot-the-diff method requires a reference set and a test set. The reference set represents the baseline distribution of the data, while the test set is used to detect any drift. The distance between the histogram vectors of the reference set and the test set is calculated and ranked for each flow feature. The top k-th features with the largest distances are identified as candidates for the occurrence of drift. However, drift detection is often impeded by the inadequate determination of thresholds for k.

Furthermore, a visualization technique was employed to identify instances of drift in QQ Music traffic. Network traffic differs from images and text, making it difficult for human perception. The t-distributed stochastic neighbor embedding (t-SNE) approach was chosen to analyze the network traffic at varying time intervals. Fig. 5a illustrates the daily collected QQ Music network traffic, revealing an overlap of the two data types. In contrast, Fig. 5b displays the monthly collected QQ Music network traffic, where a clear distinction between the left and right sides of the diagram can be observed.

5 Solutions to Network Traffic Data Distribution Shift

5.1 Common Approaches

There are three common ways to cope with the data distribution shift problem: 1) Continuously supplementing new data for model retraining. This requires the training dataset to be large enough for the AI-TC model to learn the distribution as comprehensively as possible. However, this is often difficult to achieve in practical applications because of the enormous size of the network traffic. The massive traffic dataset will take up a great deal of storage resource, which also sharply increases the cost of model training. So, the academic community is also constantly looking for new methods to make the model cope with the network data shift problem at a limited cost. 2) Retraining the model using newly labeled data. The different training methods are divided into two ways: retraining the model on both new and old data (also called stateless training) and using only new data for continuous training on the existing model (known as stateful training or fine-tuning). Regardless of the paradigm selected, two core questions need to be clarified: how to choose between stateful and stateless training and how to start a new training. 3) Domain adaptation. To solve the problems of the previous two approaches, researchers have proposed a new area of research in machine learning. This method allows the model to learn the target distribution without new labels. It often uses representation learning methods combined with unsupervised learning to enable the model to learn invariant features of the data, thereby effectively addressing data distribution shift[15–16].

5.2 Instance-Based Adaptive Methods

Instance-based adaptation is a process in domain adaptation that aims to minimize the bias among data distributions by reweighting source-labeled data according to the target risk. It assumes that the source and target distributions differ only in their marginal distributions, while the posterior distribution remains constant, called covariate bias. The problem is difficult to solve without labeled data in the target domain. To solve this problem, an importance weighting method is used to compensate for the bias by reweighting the samples in the source domain according to the ratio of the densities in the tar-

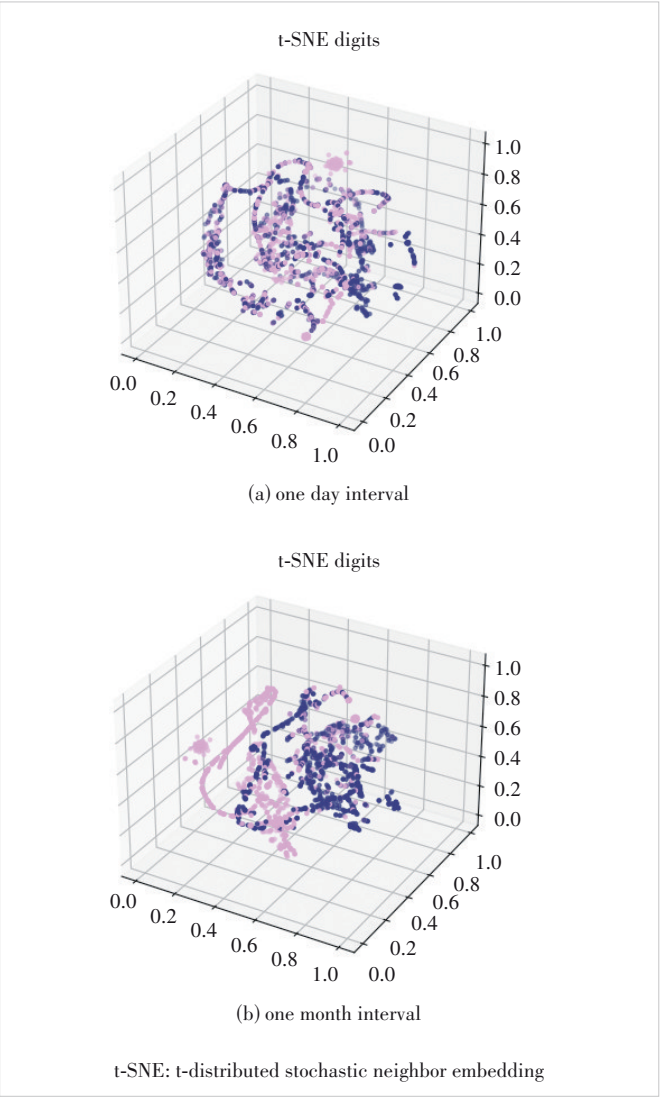


Figure 6. Visualisation of network traffic characteristics for different time intervals

get and source domains. This can be achieved by methods such as kernel mean matching (KMM) and the Kullback-Leibler importance estimation procedure (KLIEP). KMM uses the MMD in the Reproducing Kernel Hilbert Space (RKHS) to estimate the weights by minimizing the difference between the domains. In contrast, the KLIEP uses the KL difference between the target and weighted source distributions to estimate the density ratio directly.

5.3 Feature-Based Adaptive Methods

Feature-based adaptation is a domain adaptation process that aims to map source data to target data by learning transformations that extract invariant feature representations across domains. The method transforms the original features into a new feature space and then minimizes the gap among domains while preserving the underlying structure of the original data. Current feature-based adaptive methods mainly include the following three methods.

1) Subspace-based methods: This approach aims to discover common intermediate representations shared among domains by constructing low-dimensional subspaces for each domain and reducing their differences. Techniques such as PCA can be used to construct subspaces and measure domain offsets.

2) Transformation-based methods: This approach transforms the original features into a new representation to minimize the differences between the marginal and conditional distributions while preserving the structure and characteristics of the original data. Differences among domains are measured using techniques such as MMD and Kullback-Leibler Divergence (KL-Divergence).

3) Reconstruction-based methods: This approach aims to reduce the differences between domain distributions by employing an intermediary feature representation to reconstruct samples. It entails acquiring a linear projection matrix that converts source samples into intermediate representations and subsequently aligns them with the samples present in the target domain.

5.4 Deep Domain Adaptive Methods

Deep domain adaptive methods utilize neural networks (e.g., CNN, VAE, GAN) to close the gap among domains and adjust the distributions between the source and target domains. These methods typically involve training two networks: one for the source data and the other for the target data. These two networks can learn the representation features of the two types of data to minimize the inter-domain differences. Current approaches based on deep domain adaptation are divided into three main directions:

1) The variance adaptive approach uses an adaptive layer in the domain adaptation to match the marginal distribution across domains, aiming to adjust the bias in the distributions.

2) The reconstructive adaptive method adopts an autoencoder to adjust the differences across domains. The autoencoder is trained on samples from one source domain and then adjusts and reconstructs samples from another domain. By minimizing the reconstruction error between the source and target domains, the autoencoder can learn to extract invariant features across domains.

3) Adversarial domain adaptation uses adversarial learning to extract domain-invariant features by minimizing distributional differences among domains.

6 Current Challenges and Future Research Directions

Both continuous learning and domain adaptation have improved network traffic classification in real-world environments to some extent, allowing models to be deployed in such scenarios. However, current techniques still suffer from the following problems:

1) Network data mobility: The characteristics of network traffic change with time, location, network topology, etc. Continuous learning requires timely access to new data, while the domain adaptive model needs to adapt to dynamic changes in domain differences in a timely manner.

2) Application domain shift: Application version updates or business upgrades will most likely lead to significant differences in traffic characteristics. Meanwhile, the features previously found by the domain adaptive models are no longer similar. How can domain shift be detected in open scenarios?

3) Lack of labeled data: The challenge of annotating network data remains persistent and unresolved. Moreover, the presence of imbalanced samples within various categories of network traffic significantly undermines the precision of the continuous learning process.

4) Real-time requirements: Network traffic classification requires rapid response to identify and address real-time cyber threats, so models must be trained and inferred under real-time requirements. When models are then deployed, few studies have considered how network traffic is pre-processed and classified in real time.

5) Model complexity and computational resources: Models used in real-world scenarios are often in a state of failure and retraining due to the extremely large number of web applications available today. This situation greatly consumes computational resources and significantly increases model complexity.

Future researchers should focus on the following directions:

1) Combining continuous learning with dynamic domain adaptive methods to address data trafficability issues;

2) Designing systems that enable end-to-end continuous learning, including model failure feedback, shift detection, model updating, and real-time application, to ensure the sustained effectiveness of the model;

3) Enhancing the rapid response capability in real-time network environments and designing efficient model update and inference strategies to meet the requirements of real-

time scenarios.

4) Investigating how to reduce model complexity and improve computational efficiency so that the model can be applied in practical scenarios;

5) Trying to use small samples or unsupervised learning on continuous learning to improve the quality of data collection and labeling and reduce reliance on labeled data.

7 Conclusions

In this paper, we propose a generic AI-driven network traffic classification workflow and design a traffic data and feature engineering framework in detail. We thoroughly study the concepts and causes of network traffic data distribution shift, summarize the existing research progress on the data distribution shift problem, and then propose corresponding shift detection methods. Considering the current shift detection techniques, we analyze the effective means to address such shift and propose the domain adaptive method applicable to AI-TC to solve the data distribution shift problem. Finally, the current challenges and future research directions of AI-TC in data distribution shift detection and continuous learning are proposed from the requirements of data labeling, model complexity, and real-time performance.

References

[1] Wang H Z, Liu J W. Research status and key technologies of network endogenous security [J]. ZTE technology journal, 2022, 167(6): 2 – 11. DOI: 10.12142/ZTETJ.202206002

[2] Lu H, Chen Y, Lou D. 5G/5G-Advanced/6G access network security technology evolution and endogenous security [J]. ZTE technology journal, 2022, 167(6): 85 – 94. DOI: 10.12142/ZTETJ.202206014

[3] Rezaei S, Liu X. Deep learning for encrypted traffic classification: an overview [J]. IEEE communications magazine, 2019, 57(5): 76 – 81. DOI: 10.1109/MCOM.2019.1800819

[4] Wang P, Chen X J, Ye F, et al. A survey of techniques for mobile service encrypted traffic classification using deep learning [J]. IEEE access, 2019, 7: 54024 – 54033. DOI: 10.1109/ACCESS.2019.2912787

[5] Aceto G, Ciuonzo D, Montieri A, et al. Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges [J]. IEEE transactions on network and service management, 2019, 16(2): 445 – 458. DOI: 10.1109/TNSM.2019.2899085

[6] Aceto G, Ciuonzo D, Montieri A, et al. Mobile encrypted traffic classification using deep learning [C]//Proceedings of Network Traffic Measurement and Analysis Conference (TMA). IEEE, 2018: 1 – 8. DOI: 10.23919/TMA.2018.8506563

[7] Wang P, Ye F, Chen X J, et al. Datanet: deep learning based encrypted network traffic classification in SDN home gateway [J]. IEEE access, 2018, 6: 55380 – 55391. DOI: 10.1109/ACCESS.2018.2872430

[8] Wang P, Li S H, Ye F, et al. PacketCGAN: exploratory study of class imbalance for encrypted traffic classification using CGAN [C]//International Conference on Communications (ICC). IEEE, 2020: 1 – 7. DOI: 10.1109/icc40277.2020.9148946

[9] Wang P, Wang Z X, Ye F, et al. ByteSGAN: a semi-supervised generative adversarial network for encrypted traffic classification in SDN Edge Gateway [J]. Computer networks, 2021, 200: 108535. DOI: 10.1016/j.comnet.2021.108535

[10] Wang Z X, Wang P, Zhou X K, et al. FLOWGAN: unbalanced network encrypted traffic identification method based on GAN [C]//IEEE International Conference on Big Data and Cloud Computing (BdCloud). IEEE, 2019: 975 – 983. DOI: 10.1109/ispa-bdcloud-sustaincom-socialcom48970.2019.00141

[11] Wang Y, Wang P, Wang Z X, et al. Evaluation of feature selection on network traffic classification [C]//IEEE International Conference on Big Data and Cloud Computing (BdCloud). IEEE, 2021: 813 – 818. DOI: 10.1109/dasc-picom-cbdcom-cyberscitech52372.2021.00135

[12] Lipton Z, Wang Y X, Smola A. Detecting and correcting for label shift with black box predictors [C]//International conference on machine learning. PMLR, 2018: 3122 – 3130. DOI: 10.48550/arXiv.1802.03646

[13] Gretton A, Borgwardt K M, Rasch M J, et al. A kernel two-sample test [J]. The Journal of machine learning research, 2012, 13(1): 723 – 773. DOI: 10.5555/2188385.2188410

[14] Liu F, Xu W, Lu J, et al. Learning deep kernels for non-parametric two-sample tests [C]//International conference on machine learning. PMLR, 2020: 6316 – 6326. DOI: 10.48550/arXiv.2002.09116

[15] Zhang K, Schölkopf B, Muandet K, et al. Domain adaptation under target and conditional shift [C]//International conference on machine learning. PMLR, 2013: 819 – 827. DOI: 10.5555/3042817.3043028

[16] Zhao H, Des Combes R T, Zhang K, et al. On learning invariant representations for domain adaptation [C]//International conference on machine learning. PMLR, 2019: 7523 – 7532. DOI: 10.48550/arXiv.1905.12013

Biographies

Zhao Jianchao is with the Cable Products Business Department, ZTE Corporation. He is engaged in the development and delivery of wireline and cable network solutions, with a focus on product implementation, service integration, and system deployment. His work involves coordinating product requirements with network operations and supporting the deployment of large-scale wireline network services. His professional interests include wireline network solutions, service delivery optimization, and operational support for carrier networks.

Geng Zhaosen is with the Cable Products Business Department, ZTE Corporation. He is a member of the FM Product Team, focusing on the planning, operation, and management of wireline and cable network products. His work involves network service deployment, traffic management, and operational optimization in large-scale wireline networks. His research interests include wireline product planning, network operations, and data-driven network management.

Li Zeyi is currently pursuing his PhD degree in cyberspace security at Nanjing University of Posts and Telecommunications, China. His research interests include network security, anomaly detection, and deep packet inspection.

Wang Pan (wangpan@njupt.edu.cn) received his BS, MS, and PhD degrees in electrical and computer engineering from Nanjing University of Posts and Telecommunications, China in 2001, 2004, and 2013, respectively, where he is currently a full professor. His research interests include AI-powered networking and security in B5G, 6G, IoT, Smart Grid, CFN, and AI-enabled big data analysis. From 2017 to 2018, he was a visiting scholar at the Department of Electrical and Computer Engineering, University of Dayton, USA. He served as a TPC member of IEEE CyberSciTech Congress. He is also a reviewer for several journals, such as *IEEE Transaction on Network and Service Management*, *IEEE Internet of Things Journal*, *Computer and Security*, and *Big Data Research*.