SIMCMTC: A SIMILARITY COMPARISON-DRIVEN METHOD FOR MOBILE APP ENCRYPTED TRAFFIC CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Classification of encrypted mobile-app traffic faces three intertwined challenges in open, dynamic environments: label contamination from third-party SDKs, traffic drift caused by app updates, and open-set classes that training cannot cover. We propose SimCMTC, a similarity comparison-driven framework that reformulates classification as a "retrieval + comparison analysis" task. Its core is CMTE, a general embedding model for encrypted mobile-app traffic that learns reusable TCP session representations through large-scale self-supervised pretraining. For unknown traffic, SimCMTC retrieves the most similar labeled samples from a reference vector store and jointly considers vector similarity and nDPI metadata similarity to yield interpretable, open-set-aware predictions. Across five datasets spanning the mobile Internet of China, Iran, and Iraq, CMTE achieves the best closed-set accuracy and F1, and attains Acc/F1 of 0.910/0.898 on TENAPP under fine-tuning-free Top-1 retrieval, approaching the strongest supervised baseline.

1 INTRODUCTION

Encrypted traffic classification is a fundamental task in network management and security (Han et al., 2026). In mobile app traffic, port disguise and encrypted payloads render plaintext-based methods such as port-number identification and DPI ineffective (Doroud et al., 2022); supervised deep learning methods based on the closed-set assumption, while improving expressive power, face three fundamental limitations in real scenarios (Enisoglu & Inan, 2026): third-party SDKs cause label contamination (Zhao et al., 2019); app updates induce significant drift in traffic data (Chen et al., 2025); and the real network is an open-set environment in which closed-set models cannot be deployed (Luo et al., 2026). The key challenge, then, is how to relax the closed-set supervised assumption and achieve robust classification of mobile app encrypted traffic in a realistic network environment where label contamination, traffic drift, and open-set classes coexist.

To this end, we propose the SimCMTC framework, which reformulates the classification task as a "retrieval + comparison analysis" task: (i) SimCMTC learns general semantic representations of TCP sessions from large-scale unlabeled traffic, yielding the encrypted mobile-app traffic embedding model CMTE (En**C**rypted **M**obile **T**raffic **E**mbedding Model); (ii) it uses CMTE to embed labeled traffic and stores the resulting embedding vectors together with their class labels and nDPI metadata in a reference vector store; and (iii) for unknown traffic, it uses an agent to make decisions. Specifically, TCP sessions that cannot be directly determined are first embedded, and similar samples are retrieved from the reference vector store based on the embedding vector. Finally, the classification result, together with the decision rationale, is produced by combining the vector similarity and the nDPI metadata similarity with the reference samples in the retrieval results. The main contributions are as follows:

- **Methodology**: We propose SimCMTC, which models mobile app TCP encrypted traffic classification as a retrieval-based comparison analysis task, designs the SimilarityDPI skill, and achieves interpretable decisions through an agent while supporting unknown-class recognition.
- **Model**: We design a general encrypted traffic embedding model CMTE, which uses ModernBERT as the backbone, introduces a visual encoder to model the temporal features of TCP session traffic, and adopts MLM, MAE reconstruction, and debiased MoCo contrastive learning for self-supervised training, thereby enabling traffic embedding.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **Experiments**: We validate on mobile Internet traffic datasets from China, Iran, and Iraq. Under the closed-set fine-tuning setting, CMTE achieves the best classification results on all five datasets; under the fine-tuning-free setting that directly uses the training set as the reference vector store for Top-1 retrieval, it attains Acc/F1 of 0.910/0.898 on the TENAPP dataset, approaching the strongest supervised baseline.

2 Related Work

2.1 Encrypted Traffic Classification

Early methods relied on port numbers and DPI, but port disguise and payload encryption render them largely ineffective (Moore & Papagiannaki, 2005; Aceto et al., 2010; Doroud et al., 2022); traditional machine learning relies on statistical features such as packet size and inter-arrival time and has limited generalization (Shi et al., 2017); deep learning methods (Shapira & Shavitt, 2021; Lin et al., 2022; Zhao et al., 2023a; Zhou et al., 2025) achieve end-to-end feature learning but are mostly oriented toward closed-set classification, depend on annotation quality, and have difficulty coping with traffic drift caused by app updates (Chen et al., 2025). Recently, related work has also focused on retrieval-based encrypted traffic classification: CBR (Zion et al., 2026) provides a retrieval paradigm but does not deeply explore the embedding model, and STAR (Cheng et al., 2025) is only adapted to website fingerprinting. Our work differs from these methods in that SimCMTC does not directly train a classifier. Instead, it learns general semantic representations and classifies via retrieval coupled with comparison analysis, which avoids the constraints of training-label contamination and the closed-set assumption.

2.2 Self-Supervised Learning and Contrastive Learning

Self-supervised learning does not rely on manual annotation; instead, it constructs pretraining tasks from the unlabeled data itself. BERT (Devlin et al., 2019) performs self-supervised pretraining with MLM and NSP, but NSP tends to learn degenerate solutions; RoBERTa (Liu et al., 2019) removes it, and ALBERT (Lan et al., 2020) replaces it with sentence order prediction. RetroMAE (Xiao et al., 2022) uses an encoder–decoder architecture to reconstruct sequences from partial tokens and a sentence vector, but it lacks explicit semantic contrast and thus has insufficient fine-grained discrimination. Contrastive learning enhances discriminability through positive and negative samples; self-supervised contrastive learning constructs positive samples via sample transformations and treats other samples as negatives. For positive-pair construction, SimCSE (Gao et al., 2021) builds positive pairs using dropout, and ESimCSE (Wu et al., 2022) builds positive pairs using word repetition. For negative-sample handling, SimCLR (Chen et al., 2020) uses in-batch data as negatives and MoCo (He et al., 2020) maintains a queue of historical negatives. Self-supervised contrastive learning has no label information, so pseudo-negative samples exist and debiasing is required: DCL (Chuang et al., 2020) requires knowledge of the negative distribution, FNC (Chiu et al., 2024) dynamically eliminates false negatives through clustering, and SwAV (Caron et al., 2020) avoids negative samples entirely through online clustering and swapped prediction over cluster assignments. CMTE jointly adopts three strategies—MLM, MAE, and contrastive learning—which not only preserve robust reconstructive representations but also enhance discriminability and mitigate dimensional collapse. To address pseudo-negative samples, we introduce a simple yet effective similarity-threshold debiasing strategy that directly masks out negative samples whose similarity to the anchor exceeds a pre-defined threshold, without relying on clustering or distributional assumptions.

3 Method

3.1 Problem Definition and System Architecture

Problem formulation. Given a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of TCP sessions from unknown traffic, where each session s_i is a sequence of packets that may belong to a set of known classes $\mathcal{C} = \{c_1, \ldots, c_k\}$ or to new classes not seen during training (denoted c_{unknown}). The goal of traffic classification is to predict, for each session s_i, its class $\hat{c}_i \in \mathcal{C} \cup \{c_{\text{unknown}}\}$ and to provide an interpretable decision rationale. Unlike typical closed-set classification $f : \mathcal{S} \to \mathcal{C}$, SimCMTC maps each session to a d-dimensional semantic vector via a representation mapping $\phi : \mathcal{S} \to \mathbb{R}^d$, and then performs retrieval

and comparison based on the cosine similarity $\mathrm{sim}(\phi(s_i), \phi(s'_j))$ against a reference vector store $\mathcal{R} = \{(\phi(s'_j), c_j, \mathrm{nDPI}_j)\}$, where $c_j \in \mathcal{C}$ and nDPI_j denote the class label and nDPI metadata of the reference sample s'_j, thereby covering both known classes and the unknown class c_{unknown}.

System architecture. SimCMTC reformulates the classification pipeline as the two-stage "reference vector store construction + retrieval and comparison-based classification," as shown in Figure 1.

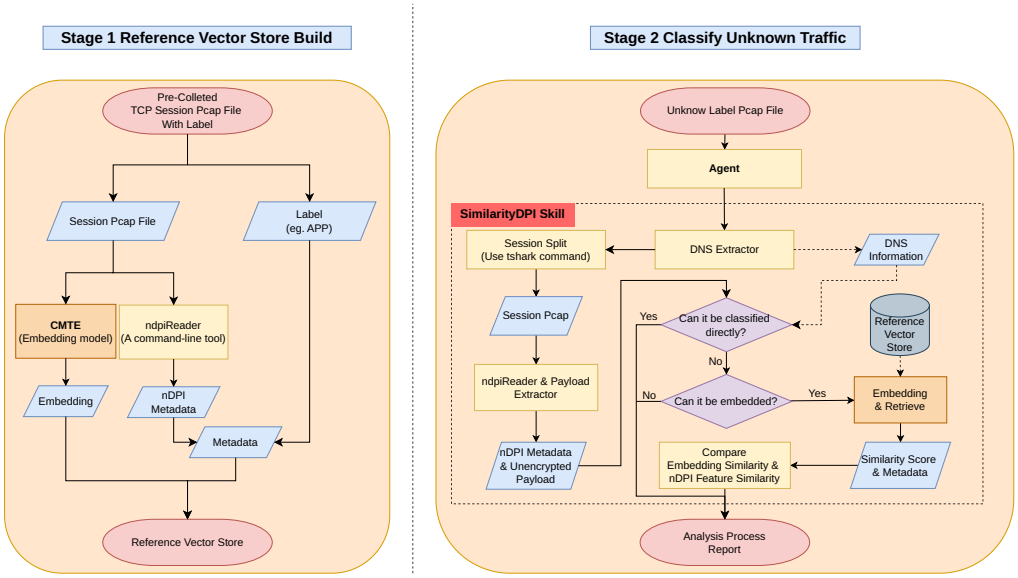


Figure 1: System architecture of SimCMTC.

SimCMTC consists of a construction stage and an inference stage. **Construction stage**: for manually collected and annotated TCP encrypted traffic, CMTE generates high-dimensional semantic embeddings, which are stored together with annotations and nDPI metadata (exported by the ndpiReader tool) in a vector store, thereby constructing an encrypted traffic reference vector store. **Inference stage**: an unknown pcap is passed to an agent configured with the SimilarityDPI skill for autonomous classification. DNS information is first extracted, and the traffic is split into session-level pcaps by TCP/UDP sessions; nDPI metadata is extracted per session. If the traffic is unencrypted, the payload content is parsed directly, and if the class can be determined by combining DNS information, the classification result is returned immediately. Otherwise, if the session meets the CMTE input requirements, the embedding service is called to obtain the embedding, and similar traffic of multiple classes (cosine similarity, class labels, nDPI metadata) is retrieved from the reference vector store. The agent then performs a side-by-side comparison between the unknown session traffic and the retrieval results, and finally produces an analysis process and classification report by combining vector similarity and nDPI metadata similarity, thereby achieving interpretable classification. Appendix E presents a complete example of one full execution of this skill.

3.2 The CMTE Embedding Model

CMTE (En**C**rypted **M**obile **T**raffic **E**mbedding Model) is the core component of the SimCMTC framework; it is responsible for mapping encrypted traffic into a high-dimensional semantic embedding space to support similarity retrieval and comparison analysis. Its overall training architecture is shown in Figure 2.

3.2.1 Training Data Preparation

CMTE is trained on the **NUDT-Mobile** dataset (Zhao et al., 2023b), which is collected at the IP layer and contains the traffic of more than 350 mainstream Chinese Internet mobile apps from 2020, totaling over 600GB. In the preprocessing stage, session pcaps with a packet count of at least 10

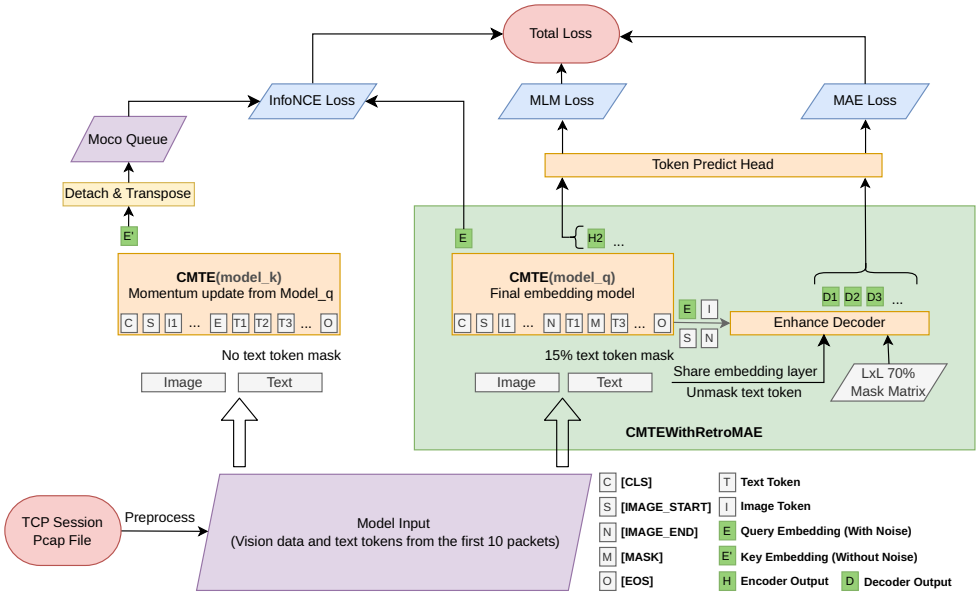


Figure 2: The multimodal training architecture of the CMTE model.

are exported and duplicate-named samples are removed, yielding 5,780,899 samples, from which 102,400 samples are randomly drawn as the **NUDT-Mobile-mini** dataset. For each pcap, the header fields of the first 10 packets that carry feature information are exported, without involving encrypted payload content or IP address information. In this paper, double quotes, newlines, and spaces are removed from the JSON format, field names and values are separated by colons, fields by commas, and packets by semicolons, as shown below:

```
frame.encap_type:7,frame.time_delta:0.000000000,frame.time_relative:0.000000000,
frame.number:1,frame.len:60,frame.protocols:raw:ip:tcp,ip.version:4,
ip.hdr_len:20,ip.dsfield:0x00,ip.len:60,ip.id:0x99ed,ip.frag_offset:0,ip.proto:6,
tcp.srcport:56863,tcp.dstport:443,tcp.stream:0,tcp.completeness:0,tcp.len:0,...;
frame.encap_type:7,frame.time_delta:0.277840000,frame.time_relative:0.277840000,
frame.number:2,frame.len:60,frame.protocols:raw:ip:tcp,ip.version:4...
```

For packets within the first 30 seconds, the packet size and packet arrival time are exported, and two-channel image data of size 224×224 are generated, where the two channels correspond to the client-to-server and server-to-client directions of the session. The image y-axis represents packet size and the x-axis represents packet arrival time. The x-axis is normalized linearly, while the y-axis is processed with a logarithm and then normalized; the packet size is clipped to a maximum of 10,000 bytes and a minimum of 40 bytes, the smallest possible IP-layer TCP packet. The position formulas are as follows:

$$X = \left\lfloor \frac{\text{packet_arrive_time}}{30} \times 223 + 0.5 \right\rfloor \tag{1}$$

$$Y = \left\lfloor \frac{\log(1 + \max(\min(\text{packet_size}, 10000), 40)) - \log(1 + 40)}{\log(1 + 10000) - \log(1 + 40)} \times 223 + 0.5 \right\rfloor \tag{2}$$

Each packet is mapped to the pixel at its computed position (X, Y), and the value of each pixel is the number of packets that fall into that cell.

3.2.2 Tokenizer Design and Efficiency Validation

To improve training efficiency, a dedicated pre-tokenizer is designed for the exported header-field data. A predefined vocabulary built from double-byte (16-bit) code points is adopted, supplemented with some printable ASCII characters, the keys and enumerable values of the tshark-exported fields, and special tokens, finally constructing a tokenizer vocabulary containing 65,708 tokens.

The experiments in Appendix A show that the average token encoding length of the CMTE tokenizer is about 45% of that of the Qwen3-VL-Embedding-2B tokenizer, cutting the average token count to less than half and thereby more than halving the encoding cost; according to the analysis of the training data, token truncation basically does not occur in the training samples (the truncation rate is only about 0.042%), so the complete sentence semantics can be preserved.

3.2.3 Model Architecture

CMTE uses ModernBERT (Warner et al., 2025) as the base model and adds a lightweight visual encoder to encode image information; both the visual encoder and the base model are randomly initialized and trained together. The visual encoder takes as input the two-channel image (one channel per traffic direction) whose x-axis encodes packet arrival time and whose y-axis encodes packet size within 30 seconds, and the final hidden-state output is mapped by a linear layer into the input word-vector space of ModernBERT, concatenated with the embeddings of the text tokens, and the embeddings of special tokens such as [CLS] are added to the sequence to jointly form the input of ModernBERT. The model takes the L2-normalized output of the first token ([CLS]) in the last layer as the embedding of the session traffic.

Since CMTE is a multimodal model and ModernBERT's rotary position encoding cannot encode the width/height information of images, we adopt interleaved multimodal rotary position encoding (MRoPE-I) (Huang et al., 2026) to replace the position encoding in ModernBERT, thereby modeling the token positions and width/height information of each modality.

3.2.4 Self-Supervised and Contrastive Learning Training Strategy

Unlike previous supervised learning schemes, our training set does not require manual annotation, because mobile app encrypted traffic has two distinctive properties: first, **SDK contamination** renders annotation inaccurate (multiple apps share/mutually call third-party SDKs); second, **traffic drift**, in particular the feature updates of emerging apps, makes their traffic distribution evolve continuously, so annotated data can hardly cover all scenarios. Therefore, we adopt a joint training strategy combining self-supervised MAE and self-supervised contrastive learning, and the total loss consists of three parts:

$$\mathcal{L} = \lambda_{\text{MLM}}\,\mathcal{L}_{\text{MLM}} + \lambda_{\text{MAE}}\,\mathcal{L}_{\text{MAE}} + \lambda_{\text{MoCo}}\,\mathcal{L}_{\text{MoCo}}, \tag{3}$$

where the three terms correspond to the MLM loss, the MAE sentence reconstruction loss, and the debiased MoCo contrastive loss, respectively. The detailed objective functions of the three losses, the details of EnhanceDecoder, and the end-to-end training pseudocode are given in Appendix B.

MLM and MAE. The MLM task randomly masks a proportion of the input token sequence, encodes it with ModernBERT, and then predicts the original tokens corresponding to the masked tokens, enabling the model to learn rich word-level semantic representations. The MAE task adopts the RetroMAE paradigm: ModernBERT encodes the masked sentence to obtain a sentence vector, which is then passed through a lightweight enhanced decoding module to reconstruct the full text segment under a restricted context (only 30% of the context information and the complete sentence vector); the reconstruction loss $\mathcal{L}_{\text{MAE}}$ forces the model to condense a semantically complete sentence-vector representation, thereby producing the traffic embedding.

Debiased MoCo contrastive learning. Constrained by GPU memory and computational cost, we adopt the MoCo framework to maintain a dynamically updated **momentum contrastive learning queue (MoCo Queue)**: the query $\mathbf{q}$ is encoded from the original sample after masking and dropout, while the positive sample $\mathbf{k}^+$ is encoded from the same original sample without masking and without dropout in the momentum encoder. After each forward pass, the batch output of the momentum encoder is added to the momentum contrastive learning queue, where the other keys serve as negative

samples; then the similarity with the query is computed, and for negative samples whose similarity exceeds the similarity threshold (i.e., potential pseudo-negatives), the corresponding similarity is set to $-\infty$ for debiasing purposes. Finally, the InfoNCE loss is computed directly from the similarity matrix via cross-entropy.

3.3 Skill Design

Traditional traffic classification models can only output a classification result and a confidence score, whereas SimCMTC can provide finer-grained classification rationales and interpretability by comparing the vector similarity and the nDPI metadata similarity between the retrieval results and the sample to be classified, where the latter is a field-wise comparison of the DPI metadata (nDPI protocol, TLS version, packet/byte statistics, etc.) exported by ndpiReader. We encapsulate the entire decision process into a skill, through which the agent makes autonomous decisions according to the skill description (see Section 3.1), and finally generates a classification report, thereby enabling interpretable classification of encrypted traffic and supporting open-set recognition of new/unknown classes. Appendix E provides the output report of one complete execution.

4 Experiments

4.1 Experimental Setup

Datasets. To comprehensively evaluate the method and cover different network scales and environments, we validate on five datasets spanning the mobile Internet environments of China, Iran, and Iraq. The detailed information of each dataset is summarized (see Table 5) in Appendix C; the comparison experiments keep the training and test sets identical across methods.

Evaluation metrics. Accuracy and macro-averaged F1 are adopted as the primary metrics: accuracy reflects the overall proportion of correct classifications, and F1 balances precision and recall under imbalanced class distributions.

Comparison methods. We compare against three open-source closed-set classification models: **ET-Bert** (Lin et al., 2022), **YaTC** (Zhao et al., 2023a), and **TrafficFormer** (Zhou et al., 2025). All three are fine-tuned on their respective pretrained models: YaTC is fine-tuned for 200 epochs, and TrafficFormer and ET-Bert are each fine-tuned for 10 epochs, taking the best test result during fine-tuning to ensure a fair comparison. TrafficFormer involves data augmentation, and we compare both the non-augmented and $5\times$-augmented versions. Implementation details of CMTE model training are given in Appendix B, Section B.5.

4.2 Main Results

We first compare CMTE as a closed-set classifier: a linear layer is appended after the CLS output of CMTE to map the pre-L2-normalized embedding to classes, and the CMTE backbone and classification head are fine-tuned for 10 epochs, taking the best test result during fine-tuning. In addition, to validate the retrieval capability of the embedding model, the pretrained CMTE embedding model is used directly for retrieval, with the training set as the reference data and the class of the Top-1 result taken as the classification result (CMTE(Retrieve + Top-1)); this scheme requires no fine-tuning. Furthermore, we report the class-wise Top-5 Recall@5 hit rate (CMTE(Retrieve + class-wise Top-5)), i.e., whether the true class appears among the top-5 nearest candidates, which is better aligned with the agent's retrieval-and-comparison strategy than single-label accuracy. Table 1 reports the Acc/F1 of each model, and the Recall@5 of the class-wise Top-5 retrieval row, on the five datasets.

Closed-set classification performance. On closed-set tests, CMTE achieves the best results on all datasets by appending only a simple linear layer and fine-tuning for 10 epochs. These results show that the general semantic representations learned by CMTE through self-supervised MAE and self-supervised contrastive learning are highly transferable.

Cross-environment generalization. In the ITC-Net-MingledApp experiment, with traffic from Tehran and Qom in Iran as training data and traffic from Iraq as test data, TrafficFormer performs poorly, generalizing weakly across network environments. Meanwhile, the fine-tuned CMTE with

Table 1: Comparison of main results (Acc / F1; Recall@5 for the class-wise Top-5 retrieval row)

Model	Dataset (Acc / F1)				
	CHNAPP	CrossNet2021	Blend-60	MingledApp	TENAPP
ET-Bert	0.953/0.943	0.409/0.265	0.275/0.223	-*	0.759/0.634
YaTC	0.963/0.962	0.335/0.304	0.693/0.691	-*	0.951/0.952
TrafficFormer	0.715/0.691	0.404/0.270	0.414/0.428	0.184/0.216	0.853/0.798
TrafficFormer (5×-aug)	0.739/0.694	0.422/0.308	0.408/0.420	0.217/0.263	0.849/0.780
CMTE (Fine-tune + head)	**0.976/0.975**	**0.498/0.489**	**0.715/0.724**	**0.730/0.701**	**0.972/0.965**
CMTE (Retrieve + Top-1)	0.681/0.629	0.204/0.163	0.450/0.445	0.278/0.379	0.910/0.898
CMTE (Retrieve + class-wise Top-5)†	0.990	0.647	0.675	0.622	0.996

*On ITC-Net-MingledApp the training and test sets have different class counts (36 vs. 23), which offical ET-Bert and YaTC code do not support. †Reports Recall@5 (the retrieval hit rate of the true class among the class-wise top-5 candidates).

a classification head shows a clear advantage, maintaining strong cross-environment and cross-data generalization.

Fine-tuning-free retrieval. When using the training set as the reference dataset and directly retrieving Top-1 with the CMTE embedding model pretrained on Chinese mobile Internet data, the model still achieves strong results on the TENAPP dataset. The retrieval performance is affected by the scale of the reference vector store. CrossNet2021 and CHNAPP, though both Chinese mobile Internet datasets, contain too few traffic samples in their reference sets to fully cover the traffic patterns of the apps, thereby leading to suboptimal retrieval results. In contrast, with a larger reference set (e.g., TENAPP), direct Top-1 retrieval already significantly outperforms the fine-tuned ET-Bert and TrafficFormer. Moreover, by updating the reference vector store in a timely manner, SimCMTC can address the traffic drift problem in a fine-tuning-free manner. Furthermore, the class-wise Top-5 Recall@5 is consistently much higher than the Top-1 Acc/F1 across all five datasets (e.g., 0.996 vs. 0.910 on TENAPP and 0.622 vs. 0.278 on ITC-Net-MingledApp), showing that expanding the candidate set from the single Top-1 result to the top-5 nearest neighbors substantially raises the probability of including the true class. The retrieval-and-comparison classification strategy exploits exactly this gain: the agent compares the query against the top-5 candidates rather than trusting the single Top-1 label, thereby alleviating the deficiency of pure Top-1 retrieval.

4.3 Ablation Study

To investigate the contribution of each module to model performance, the TENAPP dataset is evaluated using retrieval + Top-1. Given the large volume of the full NUDT-Mobile dataset, its subset NUDT-Mobile-mini is extracted as training samples; all experiments uniformly use 10 epochs (versus 3 epochs for the full-data model) with the other hyperparameters kept identical, and the Acc/F1 of the retrieval + Top-1 results are reported, focusing on validating three modules: the visual encoder, debiased MoCo contrastive learning, and the debiasing threshold (Table 2). Because these ablations are trained on NUDT-Mobile-mini, their absolute Acc/F1 (e.g., 0.890/0.874 for the full model) is lower than the main retrieval result on TENAPP in Table 1 (0.910/0.898), which uses the model trained on the full NUDT-Mobile dataset; the relative differences between rows are the focus of this study.

Role of the visual encoder. Removing the visual encoder degrades performance, but only to a limited extent. This indicates that the first 10 packets of a TCP session already carry fairly complete discriminative information, and the visual branch, as a complementary auxiliary signal, also contributes to some degree.

Necessity of debiasing. Compared with the debiased version, using contrastive learning **without debiasing** leads to a notable drop (Acc falls by 0.070), which demonstrates that the pseudo-negatives in the sample queue are significantly harmful to model training and that debiasing is indispensable.

Sensitivity to the debiasing threshold. Setting the debiasing threshold to 0.85 and 0.95 both cause a small drop, confirming that a threshold of 0.9 is a better compromise: too low a threshold wrongly removes useful hard negatives and weakens discriminative ability, while too high a threshold admits

Table 2: Ablation study results (retrieval + Top-1 evaluation).

Visual encoder	MoCo contrastive	Debiasing threshold	Acc / F1
✓	✓	0.9	**0.890 / 0.874**[*]
×	✓	0.9	0.875 / 0.857
✓	✓	-	0.820 / 0.783
✓	✓	0.85	0.886 / 0.869
✓	✓	0.95	0.882 / 0.865
✓	×	–	0.898 / 0.886[†]

[*]Full model (baseline) with all modules enabled: visual encoder, debiased MoCo contrastive learning, and debiasing threshold 0.9. [†]Although removing contrastive learning yields a slightly higher Acc/F1, the embedding space is *not fully expanded*: without contrastive learning the vectors undergo dimensional collapse (see the PCA analysis in Section 4.4).

more pseudo-negatives. The performance rise from keeping all (no debiasing) to 0.85 and then to 0.9 also indicates that the benefit of debiasing outweighs the loss of hard negatives.

Role of contrastive learning. After removing contrastive learning, the Acc/F1 even rises slightly, but the embedding vectors undergo dimensional collapse (see the analysis in Section 4.4), destroying the high-quality uniform representations required for accurate retrieval. The value of contrastive learning lies not in improving the single-point classification metric but in ensuring the geometric quality of the embedding space in the retrieval scenario.

4.4 In-Depth Analysis

Dimensional-collapse visualization. To reveal the influence of contrastive learning and data scale on the data distribution of the embedding space, we perform a principal component analysis of the test embeddings (Musil, 2019), as detailed in Appendix D. For the test set of TENAPP, we accumulate the normalized variance of each embedding dimension up to a threshold of 0.95; the number of dimensions required to reach this threshold directly reflects how well the embedding space exploits its dimensions. For a model trained on **NUDT-Mobile-mini**, reaching the threshold without contrastive learning requires only **42** dimensions, whereas adding debiased contrastive learning (threshold 0.9) raises this to **105** dimensions. A model trained on the full **NUDT-Mobile** requires **209** dimensions. These results show that contrastive learning effectively mitigates the dimensional-collapse problem and encourages the model to use more dimensions to encode sample information; increasing the pretraining data scale further improves dimension utilization. The underlying mechanism is as follows. Without contrastive learning, the sentence vector is optimized directly by the sentence-reconstruction objective of the enhanced decoder. In the reconstruction loss, high-frequency tokens contribute a systematically larger share because they occur frequently; to minimize the total loss, the model therefore concentrates the [CLS] embedding along the general semantic direction most correlated with these tokens. Consequently, the variance of high-frequency-related dimensions is repeatedly reinforced, while low-frequency and fine-grained semantic dimensions are relatively neglected. The representation space thus becomes anisotropic and contracts onto a narrow cone (Gao et al., 2019), reducing its effective rank and producing dimensional collapse. Contrastive learning reverses this trend: the repulsion between negative pairs pushes different samples apart, curbing the dominance of a few high-frequency semantic directions and encouraging the model to exploit more dimensions for discriminative information. A larger data scale further improves dimension utilization through richer contextual patterns.

Quality metrics of contrastive learning. The training-process curves (Figure 4) reflect three aspects. First, the positive pair similarity is relatively high and approaches 1, indicating that under 15% masking and 10% dropout noise, the MoCo framework still maps positive pairs to similar semantic regions and achieves a degree of denoising and semantic alignment, consistent with the optimization objective of the positive pairs. Second, the total loss steadily decreases, and the mean negative-sample similarity stabilizes near 0. For normalized hypersphere embeddings, if the embeddings are approximately uniformly distributed or isotropic and the representation dimensionality has not degraded, the mean cosine similarity of 16K+ negative samples should be close to 0; the observed mean inter-sample similarity of the NUDT-Mobile-mini embeddings is 0.001, supporting this uni-

formity/isotropy trend. Third, the Margin (the positive similarity minus the best negative similarity, before debiasing) can reach 0 or even become negative, suggesting that pseudo-negatives may exist in the negative list and providing indirect evidence for the necessity of debiasing. Meanwhile, the moving average of the margin before debiasing approaches 0.25, indicating that a certain cosine similarity gap is retained on average between the positive sample and the most similar negative sample, which is beneficial for retrieval.

Failure cases and analysis of limitations. Appendix F further presents a summary report for the 1635 TCP sessions in the TENAPP dataset that failed Top-1 retrieval, in which comparison analysis and evaluation were conducted using an agent combined with the SimilarityDPI skill. The results reported therein show that, with the assistance of the agent, after fusing vector similarity and nDPI metadata similarity, 70.83% of mobile app traffic can be correctly classified. Even when the reference data is contaminated by SDKs, the SimCMTC framework can still filter out some open-set traffic, which mainly originates from public SDKs, advertisements, and failed connection requests. These results verify that the SimCMTC framework is capable of open-set traffic analysis after using an agent to retrieve and comparatively analyze encrypted traffic. Although SimCMTC performs well overall, there remain notable failure scenarios worth discussing. First, when the traffic patterns of certain apps are highly similar (e.g., multiple apps sharing the same third-party SDK with identical call traces), the retrieved Top-K neighbors may be confused across classes. The dual comparison of vector similarity and nDPI metadata similarity still cannot distinguish them, at which point the classification confidence is low; second, when the reference vector store has insufficient traffic coverage for an app (e.g., too few samples or not covering all functional traffic), the retrieval results tend to be biased, as shown by the retrieval performance on CHNAPP; third, in cross-region environments, although CMTE maintains strong generalization over the supervised baselines, its retrieval accuracy remains low (0.278/0.379 Acc/F1 on ITC-Net-MingledApp), reflecting the inherent difficulty of cross-region classification under class imbalance. These limitations point to future work on improving discriminability, expanding reference-store coverage, and addressing class imbalance.

5 Conclusion

Aiming at the core challenges of label contamination, traffic drift, and open-set recognition in mobile app encrypted traffic classification, we propose the SimCMTC framework, which reformulates classification as "retrieval + comparison analysis" and designs the multimodal embedding model CMTE that combines self-supervised MAE and self-supervised debiased contrastive learning. On five datasets covering the mobile Internet environments of China, Iran, and Iraq, CMTE's closed-set classification results are better than those of other models, demonstrating strong cross-environment generalization; direct retrieval without fine-tuning still achieves Acc/F1 = 0.910/0.898 on TENAPP; the ablation and in-depth analyses validate the key role of the visual encoder, the debiasing operation, and contrastive learning in embedding-space quality. On this basis, we further apply the complete SimCMTC to the 1635 TCP session pcap files whose Top-1 retrieval failed in the TENAPP test set; the agent's final decision accuracy reaches 70.83%, which demonstrates the discriminative capability of the SimCMTC framework on open-set traffic.

Reproducibility Statement

We provide the inference and evaluation scripts and the CMTE model weights pretrained on the NUDT-Mobile dataset in the supplementary material. The preprocessing pipeline (Section 3.2.1), the end-to-end training procedure (Algorithm 1), and the complete hyperparameter settings (the "Implementation Details" subsection of Appendix B) are described in detail, together with the dataset statistics in Appendix C, enabling the reproduction of the main results reported in Table 1.

AI Use Statement

The authors used Claude Code (powered by the DeepSeek model) during the preparation of this work. The specific uses are disclosed below in accordance with the ICLR 2027 AI Use Policy.

Mandatory disclosures:

- **Methodology or experimental design:** The tool was used to assist in discussing and validating the technical approach. All final methodological decisions and experimental designs were made by the authors.
- **Method implementation:** The tool was used to generate preprocessing code for reproducing baseline methods in closed-set comparison experiments. The authors reviewed, tested, and validated all generated code.
- **Qualitative analysis and result interpretation:** The tool was used to assist in analyzing experimental data. The authors independently verified all analyses and interpretations.

Optional disclosures:

- **Literature search:** The tool was used to assist in literature search.
- **Code writing:** The tool was used to write evaluation scripts and other non-core code. The authors reviewed and tested all code.
- **Paper polishing:** The tool was used for paper polishing. The authors reviewed and edited all text.

All AI-generated or AI-assisted outputs were treated as auxiliary suggestions. The authors independently reviewed, verified, and edited all text, code, references, mathematical derivations, figures, and experimental results. AI tools were not used to fabricate citations, hide prompt injections, or generate undisclosed scientific content. The authors take full responsibility for the content of this paper, including all claims, data, analyses, and conclusions. This disclosure is consistent with the submission form.

This statement is provided in accordance with the ICLR 2027 AI Use Policy.

REFERENCES

Giuseppe Aceto, Alberto Dainotti, Walter de Donato, and Antonio Pescape. Portload: Taking the best of two worlds in traffic classification. In *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, pp. 1–5. IEEE, March 2010. doi: 10.1109/infcomw.2010.5466645.

Marziyeh Bayat, Javad Garshasbi, Mozhgan Mehdizadeh, Neda Nozari, Abolghasem Rezaei Khesal, Maryam Dokhaei, and Mehdi Teimouri. Itc-net-blend-60: a comprehensive dataset for robust network traffic classification in diverse environments. *BMC Research Notes*, 17(1), June 2024. ISSN 1756-0500. doi: 10.1186/s13104-024-06817-5.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Neural Information Processing Systems*, 2020. URL https://mlanthology.org/neurips/2020/caron2020neurips-unsupervised/.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Zihan Chen, Guang Cheng, Jinhui Li, Tian Qin, Yuyang Zhou, and Xing Luan. Drift-oriented self-evolving encrypted traffic application classification for actual network environment, 2025.

Yifei Cheng, Yujia Zhu, Baiyang Li, Xinhao Deng, Yitong Cai, Yaochen Ren, and Qingyun Liu. Star: Semantic-traffic alignment and retrieval for zero-shot https website fingerprinting, 2025.

Chi-Min Chiu, Ying-Jia Lin, and Hung-Yu Kao. *Contrastive Learning for Unsupervised Sentence Embedding with False Negative Calibration*, pp. 290–301. Springer Nature Singapore, 2024. ISBN 9789819722594. doi: 10.1007/978-981-97-2259-4_22.

Ching-Yao Chuang, Joshua W. Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased Contrastive Learning. In *Neural Information Processing Systems*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423.

Hossein Doroud, Ahmad Alaswad, and Falko Dressler. Encrypted traffic detection: Beyond the port number era. In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pp. 198–204. IEEE, 2022. doi: 10.1109/lcn53696.2022.9843432.

Ramazan Enisoglu and Burak Alp Inan. Network traffic classification from handcrafted features to foundation models: A comprehensive survey. *Computer Networks*, 287:112578, September 2026. ISSN 1389-1286. doi: 10.1016/j.comnet.2026.112578.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations (ICLR 2019)*, New Orleans, LA, USA, 2019. OpenReview.net. URL https://openreview.net/forum?id=SkEYojRqtm.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.552.

Shangbin Han, Han Zhang, Mengmeng Lu, Sifang Guo, Boyuan Tian, and Jilong Wang. A comprehensive survey on encrypted network traffic classification. *Computer Networks*, 287:112524, September 2026. ISSN 1389-1286. doi: 10.1016/j.comnet.2026.112524.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735. IEEE, June 2020. doi: 10.1109/cvpr42600.2020.00975.

Jie Huang, Xuejing Liu, Sibo Song, Ruibing Hou, Hong Chang, Junyang Lin, and Shuai Bai. Revisiting multimodal positional encoding in vision-language models. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=sCCF4ygDAw.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking, 2026.

Wenhao Li, Xiao-Yu Zhang, Huaifeng Bao, Haichao Shi, and Qiang Wang. Prograph: Robust network traffic identification with graph propagation. *IEEE/ACM Transactions on Networking*, 31(3): 1385–1399, June 2023. ISSN 1558-2566. doi: 10.1109/tnet.2022.3216603.

Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 633–642. ACM, April 2022. doi: 10.1145/3485447.3512217.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Zijian Luo, Shuang Zhao, Hongyi Chen, Ziling Wei, and Yang Wang. Decoupled dual-branch framework for open-set network traffic classification. In *2026 IEEE 51st Conference on Local Computer Networks (LCN)*, pp. 1–9. IEEE, October 2026. doi: 10.1109/lcn67947.2026.11660812.

Andrew W. Moore and Konstantina Papagiannaki. *Toward the Accurate Identification of Network Applications*, pp. 41–54. Springer Berlin Heidelberg, 2005. ISBN 9783540319665. doi: 10.1007/978-3-540-31966-5_4.

Tomáš Musil. *Examining Structure of Word Embeddings with PCA*, pp. 211–223. Springer International Publishing, 2019. ISBN 9783030279479. doi: 10.1007/978-3-030-27947-9_18.

Abolghasem Rezaei Khesal, Mohammad Sadeq Abed, Mohammad Mahdi Mohammadi, Mohammad Mahdi Mahdavirad, Ramtin Kabiri, and Mehdi Teimouri. Itc-net-mingledapp: A comprehensive dataset of mixed mobile application traffic for network traffic classification in diverse environments. *Data in Brief*, 66:112731, June 2026. ISSN 2352-3409. doi: 10.1016/j.dib.2026.112731.

Tal Shapira and Yuval Shavitt. Flowpic: A generic representation for encrypted traffic classification and applications identification. *IEEE Transactions on Network and Service Management*, 18(2): 1218–1232, June 2021. ISSN 2373-7379. doi: 10.1109/tnsm.2021.3071441.

Hongtao Shi, Hongping Li, Dan Zhang, Chaqiu Cheng, and Wei Wu. Efficient and robust feature extraction and selection for traffic classification. *Computer Networks*, 119:1–16, June 2017. ISSN 1389-1286. doi: 10.1016/j.comnet.2017.03.011.

Zihao Wang, Wei Peng, Junming Zhang, Jian Li, and Wenxin Fang. Tao-net: Two-stage adaptive ood classification network for fine-grained encrypted traffic classification. *Neurocomputing*, 679: 133170, May 2026. ISSN 0925-2312. doi: 10.1016/j.neucom.2026.133170.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.127.

Xing Wu, Chaochen Gao, Liangjun Zang, Jizhong Han, Zhongyuan Wang, and Songlin Hu. ESimCSE: Enhanced sample building method for contrastive learning of unsupervised sentence embedding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 3898–3907, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. Retromae: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 538–548. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.35.

Ruijie Zhao, Mingwei Zhan, Xianwen Deng, Yanhao Wang, Yijun Wang, Guan Gui, and Zhi Xue. Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):5420–5427, June 2023a. ISSN 2159-5399. doi: 10.1609/aaai.v37i4.25674.

Shuang Zhao, Shuhui Chen, Yipin Sun, Zhiping Cai, and Jinshu Su. Identifying known and unknown mobile application traffic using a multilevel classifier. *Security and Communication Networks*, 2019:1–11, January 2019. ISSN 1939-0122. doi: 10.1155/2019/9595081.

Shuang Zhao, Shuhui Chen, Fei Wang, Ziling Wei, Jincheng Zhong, and Jianbing Liang. A large-scale mobile traffic dataset for mobile application identification. *The Computer Journal*, 67(4): 1501–1513, August 2023b. ISSN 1460-2067. doi: 10.1093/comjnl/bxad076.

Guangmeng Zhou, Xiongwen Guo, Zhuotao Liu, Tong Li, Qi Li, and Ke Xu. Trafficformer: An efficient pre-trained model for traffic data. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 1844–1860. IEEE, May 2025. doi: 10.1109/sp61157.2025.00102.

Yehonatan Zion, Amit Dvir, Chen Hajaj, Ran Dubin, and Neriya Filber. Ultra fast encrypted network traffic classification based on classification by retrieval powered by out-of-distribution. *Computer Networks*, 289:112739, October 2026. ISSN 1389-1286. doi: 10.1016/j.comnet.2026.112739.

A Tokenizer Evaluation Results

To validate the improvement in token encoding efficiency of the dedicated tokenizer, a comparison is performed against the Qwen3-VL-Embedding-2B (Li et al., 2026) tokenizer on the 95,976 exported samples of the ITC-Net-Blend-60 training set. The encoding-length comparison is shown in Table 3.

A.1 Tokenizer Encoding Efficiency Comparison

Table 3: Comparison of token encoding lengths between the CMTE and Qwen3VL-Embedding-2B tokenizers in ITC-Net-Blend-60 training set.

Tokenizer	Average length	Max length	Min length
CMTE tokenizer	1597	5684	968
Qwen3-VL-Embedding-2B tokenizer	3512	16861	1912

A.2 Token Distribution of Pretraining Data

We also compute the data distribution after passing the NUDT-Mobile dataset through the pre-tokenizer (after overlong truncation), as shown in Table 4.

Table 4: Token distribution statistics of the pretraining dataset after the pre-tokenizer (after overlong truncation)

Count	Min length	Max length	Average length	Truncation count (ratio)
5,780,899	763	7992	1927.73	2400 (0.042%)

Since the maximum encoding length is 7992, samples with an encoded length of 7992 are statistically treated as truncated samples.

B CMTE Training Objectives and Training Algorithm

This appendix provides the formal details of the self-supervised and contrastive learning training strategy, including the objective functions of the three losses, the construction of the EnhanceDecoder, the pseudocode of the end-to-end self-supervised pretraining of CMTE, and the implementation details.

B.1 The MLM Task

Let the input sequence of a session be $\mathbf{X} = (x_1, \ldots, x_L)$, which contains special tokens (including [CLS], [IMAGE_START], [IMAGE_END], and [EOS]), visual tokens $\mathbf{I}$, and text tokens $\mathbf{T}$; the text tokens $\mathbf{T}$ are randomly masked with probability $p_{\text{MLM}} = 0.15$. After encoding, the original text tokens t_i are predicted at the masked positions $i \in \mathcal{M}$, and the MLM loss is the cross-entropy at the masked positions:

$$\mathcal{L}_{\text{MLM}} = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \log P\left(t_i \mid \mathbf{X}_{\mathcal{M}}\right), \tag{4}$$

where $\mathbf{X}_{\mathcal{M}}$ is the sequence after masking the positions in $\mathcal{M}$, which provides the model with word-level semantic supervision.

B.2 MAE Sentence Reconstruction and EnhanceDecoder

Let $\mathbf{EM}$ be the shared text-token embedding matrix, and H the hidden dimension. In the MLM task, for the masked sequence $\mathbf{X}_{\mathcal{M}}$, after encoding, the hidden state $\mathbf{H}^{(e)} = \text{Enc}(\mathbf{X}_{\mathcal{M}}) \in \mathbb{R}^{L \times H}$ is obtained, whose first token is the sentence vector $\mathbf{E} = \mathbf{H}^{(e)}_{[\text{CLS}]}$. On the decoding side, a lightweight enhanced decoding module (EnhanceDecoder, a one-layer Transformer) is introduced to reconstruct the complete sentence under a restricted context: the sentence vector $\mathbf{E}$ is expanded to the same length as the sequence (both of length L) as $\mathbf{H}_1$, and the input embeddings $\mathbf{X}_{\text{emb}}$ of the unmasked original sentence (visual tokens from the visual encoder, text tokens from the shared embedding matrix $\mathbf{EM}$, and the special-token embeddings excluding [CLS]) are obtained, with the sentence vector $\mathbf{E}$ concatenated at the head to form $\mathbf{H}_2$:

$$\mathbf{H}_1 = [\mathbf{E}; \ldots; \mathbf{E}] \in \mathbb{R}^{L \times H}, \qquad \mathbf{H}_2 = \left[\mathbf{E};\ \mathbf{X}_{\text{emb}}\right] \in \mathbb{R}^{L \times H}. \tag{5}$$

The queries in the attention computation are computed from $\mathbf{H}_1$, and the keys and values from $\mathbf{H}_2$. To force the model to extract more information from the sentence vector, the EnhanceDecoder imposes a mask $\mathbf{M}^{\text{dec}} \in \{0, 1\}^{L \times L}$ on the decoding attention (diagonal elements are masked, the first column of each row, i.e., the sentence-vector position, is retained, and the rest are sampled according to Bernoulli$(1 - p_{\text{dec}})$ with $p_{\text{dec}} = 0.7$, i.e., 30% are retained and the remaining 70% are masked), obtaining the final hidden-layer output $\mathbf{H}^{(d)}$.

The enhanced decoding module realizes, through the mask matrix $\mathbf{M}^{\text{dec}}$, the context-masking requirement for each output position i: each position can attend to only 30% of the unmasked tokens and the complete sentence vector $\mathbf{E}$ — therefore every token can participate in the computation of the sentence-reconstruction loss. Formally, by averaging the cross-entropy over all text-token positions:

$$\mathcal{L}_{\text{MAE}} = -\frac{1}{L_T} \sum_{i=1}^{L_T} \log P\Big(t_i \mid \mathbf{TH}_i^{(d)}\Big), \tag{6}$$

where L_T is the number of text tokens and $\mathbf{TH}_i^{(d)}$ is the decoder output $\mathbf{H}^{(d)}$ at the i-th text-token position (text tokens are re-indexed by their order within the sequence).

B.3 Debiased MoCo Contrastive Learning

Let f_q be the query encoder and f_k the momentum encoder, with query $\mathbf{q}_i = f_q(\mathbf{X}_i^{(q)})$ and positive sample $\mathbf{k}_i^+ = f_k(\mathbf{X}_i^{(k)})$, where $\mathbf{X}_i^{(q)}$ is the masked view of sample i (i.e., $\mathbf{X}_{\mathcal{M}}$ in Eq. (4)) and $\mathbf{X}_i^{(k)}$ is the same sample without masking. The query $\mathbf{q}_i$ is the L2-normalized [CLS] output of the query encoder (i.e., the L2-normalized sentence vector $\mathbf{E}$), and the key embedding $\mathbf{k}_i^+$ is the L2-normalized [CLS] output of the momentum encoder for the unmasked original sample. The key embedding is then added to the momentum contrastive learning queue (MoCo Queue) $\mathcal{Q} \in \mathbb{R}^{d\times N}$, with each key embedding stored as a column, so that, since both query and key embeddings are L2-normalized, the cosine similarity between each query and each sample in the queue can later be computed directly by the matrix product $\mathbf{Q}^\top \mathcal{Q}$, where $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_B] \in \mathbb{R}^{d\times B}$ is the query matrix and B is the batch size.

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\text{sim}(\mathbf{q}_i, \mathbf{k}_{\pi(i)})/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(\mathbf{q}_i, \mathbf{k}_j)/\tau)}, \tag{7}$$

where $\pi(i)$ is the queue index at which the positive sample $\mathbf{k}_i^+$ of query i is stored (so that $\mathbf{k}_{\pi(i)} = \mathbf{k}_i^+$), $\{\mathbf{k}_j\}_{j=1}^N$ are the key embeddings in the momentum contrastive learning queue, N is the queue size, τ is the temperature coefficient, and $\text{sim}(\cdot,\cdot)$ is the cosine similarity. Note that Eq. (7) is the undebiased InfoNCE loss; it is debiased below and used as $\mathcal{L}_{\text{MoCo}}$ in the total loss Eq. (3). Because, under the unlabeled condition, the queue may contain **pseudo-negative** samples that are semantically similar to or even of the same class as the query sample, we set a similarity threshold $\theta_{\text{deb}} = 0.9$ and directly set to $-\infty$ the similarity of the negative samples in the queue whose similarity to the query exceeds the threshold (i.e., potential pseudo-negatives), thereby achieving debiasing:

$$\tilde{s}_{j,i} = \begin{cases} \text{sim}(\mathbf{q}_i, \mathbf{k}_j), & j = \pi(i) \text{ or } \text{sim}(\mathbf{q}_i, \mathbf{k}_j) \le \theta_{\text{deb}}, \\ -\infty, & j \ne \pi(i) \text{ and } \text{sim}(\mathbf{q}_i, \mathbf{k}_j) > \theta_{\text{deb}}, \end{cases} \tag{8}$$

thus obtaining the debiased InfoNCE loss:

$$\mathcal{L}_{\text{MoCo}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\tilde{s}_{\pi(i),i}/\tau)}{\sum_{j=1}^{N} \exp(\tilde{s}_{j,i}/\tau)}, \tag{9}$$

This mechanism eliminates the harmful influence of pseudo-negatives without requiring additional clustering or distribution assumptions, while retaining the good sentence-vector representations already learned by the sentence reconstruction task, making the embeddings more uniformly distributed on the hypersphere and alleviating dimensional collapse.

B.4 End-to-End Training Procedure

Combining the above three subtasks, the pseudocode of the end-to-end self-supervised training of CMTE is shown in Algorithm 1.

Algorithm 1 CMTE multi-task self-supervised pretraining

Require: Large-scale unlabeled session data $\mathcal{D}$; MLM masking ratio p_{MLM}, sentence-reconstruction mask ratio p_{dec}; dropout ratio p_{drop}; temperature coefficient τ; debiasing threshold θ_{deb}; learning rate η; momentum coefficient m; queue capacity N; loss weights $\lambda_{\text{MLM}}, \lambda_{\text{MAE}}, \lambda_{\text{MoCo}}$

Ensure: The trained query encoder f_q (used as the CMTE embedding model)

Initialize the query encoder f_q and the key encoder f_k (parameters $\theta_k \leftarrow \theta_q$), set the dropout probability p_{drop} for f_q and none for f_k; initialize the momentum contrastive learning queue $\mathcal{Q}$ (MoCo Queue): randomly initialize with a normal distribution and then perform L2 normalization
for $e = 1, \ldots, E$ **do**
  for each mini-batch $\mathcal{B} \in \mathcal{D}$ **do**
    For each original pair $\{(\mathbf{X}^{(q)}, \mathbf{X}^{(k)})\}$, $\mathbf{X}^{(q)}$ is the masked view of the sample (masking ratio p_{MLM}) and $\mathbf{X}^{(k)}$ is the same sample without masking; compute the query $\mathbf{q} \leftarrow f_q\left(\mathbf{X}^{(q)}\right)$ with dropout probability p_{drop}, and the positive sample $\mathbf{k}^+ \leftarrow f_k\left(\mathbf{X}^{(k)}\right)$ without dropout
    f_q simultaneously outputs the hidden states $\mathbf{H}^{(e)}$; compute $\mathcal{L}_{\text{MLM}}$ according to equation 4
    Construct $\mathbf{H}_1, \mathbf{H}_2$ according to equation 5; in the attention computation of the EnhanceDecoder, the Query is projected from $\mathbf{H}_1$ while the Key and Value are projected from $\mathbf{H}_2$, and an attention mask matrix with masking ratio p_{dec} is applied; the output $\mathbf{H}^{(d)}$ is obtained and $\mathcal{L}_{\text{MAE}}$ is computed according to equation 6
    Transpose the current positive sample $\mathbf{k}^+$ and push it into the momentum contrastive learning queue $\mathcal{Q}$ in order, recording the index position $\pi(i)$
    According to equation 8, set to $-\infty$ the similarity of the pseudo-negatives in the queue $\mathcal{Q}$ whose similarity to the query exceeds θ_{deb}; compute $\mathcal{L}_{\text{MoCo}}$ directly with cross-entropy over all N items of the queue according to equation 9
    Compute the total loss $\mathcal{L}$ according to equation 3, and update θ_q by backpropagation
    Update the key encoder by momentum: $\theta_k \leftarrow m\,\theta_k + (1-m)\,\theta_q$
  end for
end for
return f_q

B.5 Implementation Details

CMTE is trained on the NUDT-Mobile dataset in parallel on 4 H100 GPUs, with 3 epochs, a per-GPU batch size of 4, and 4 gradient-accumulation steps; the momentum update coefficient is 0.999 and the MoCo queue size is 16,384; the learning rate is 1×10^{-4} (initial 1×10^{-7}), using linear warmup and cosine annealing, with warmup accounting for 10% of the training steps; the optimizer is AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with weight decay 0.01; the MLM masking ratio is 15%, the query dropout ratio is 10%, the MAE decoder masking ratio is 70%, the contrastive learning temperature is 0.1, and the debiasing threshold is 0.9, with the three loss coefficients being 0.2, 0.1, and 0.1, respectively; the patch size of the visual encoder is 16×16, with a fixed 224x224 input, outputting 196 visual tokens.

C Detailed Dataset Introduction

To comprehensively evaluate the effectiveness and generalization of the method and to cover different network scales and environments, we validate on five datasets spanning Chinese, international, and cross-network environments. Their detailed sources, composition, and division are as follows:

- **CHNAPP**: proposed by the TAO-Net paper (Wang et al., 2026), it collects traffic of six types of apps on the Chinese Internet: QQ Music, QQ Mail, WeChat, Weibo, Youku, and Taobao. Since this data has no ready-made division, we manually divide it into training/test sets by 9:1 for closed-set classification evaluation of mainstream Chinese apps.
- **CrossNet2021** (Li et al., 2023): collects traffic of 20 mobile apps on the Chinese Internet in two different network-environment scenarios, using scenario A as the training set and scenario B as the test set for cross-environment generalization evaluation.
- **ITC-Net-Blend-60** (Bayat et al., 2024): collected in the Iranian mobile Internet environment. It covers traffic of 60 common international apps across 5 distinct device-network scenarios, and serves to evaluate the method's recognition capability in the local Iranian device/network environment.
- **ITC-Net-MingledApp** (Rezaei Khesal et al., 2026): This dataset comprises traffic data of 36 common international apps. The data were collected across Tehran (Iran), Qom (Iran), and Iraq. The Iraqi subset is used as the test set and the Tehran and Qom subsets as the training set for cross-region generalization evaluation.
- **TENAPP**: a dataset constructed in this paper for retrieval scenarios. It was autonomously collected through an agent combined with the adb tool and the PcapDroid software. The data collection spanned from October 2025 to January 2026, and the dataset exceeds 60GB in size. It contains ten types of social-networking and short-video apps: QQ, WeChat, Bilibili, Kuaishou, Momo, Soul, Tantan, Douyin, Weibo, and Xiaohongshu (Rednote); the training and test sets are collected on different dates.

To ensure input consistency, each dataset is split by TCP sessions, and sessions with a packet count of at least 10 are extracted as training and test data, ensuring that the numbers of training and test samples are identical across models. Since the CMTE training data NUDT-Mobile is collected at the IP layer (without the Ethernet header), an additional Ethernet-header stripping step is performed on these datasets for CMTE experiments. TrafficFormer also had a small fraction of data-preprocessing failures and used data augmentation.

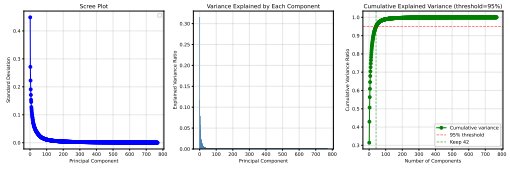
Table 5: Number of training (reference) and test samples in each dataset

Dataset	Training (reference)	Test
CHNAPP	2684	295
CrossNet2021	4128	2572
ITC-Net-Blend-60	95976	15681
ITC-Net-MingledApp	63949	12786
TENAPP	264554	18250

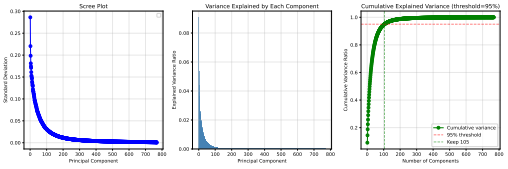
D Dimensional-Collapse Visualization, Training Curves, and Margin Distance

D.1 PCA Visualization of Dimensional Collapse

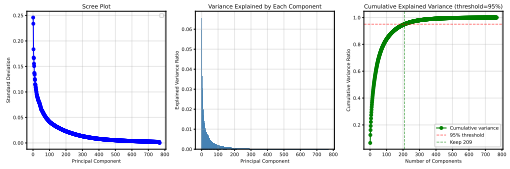
Figure 3a, 3b, and 3c respectively show the PCA results of the encoded embeddings output for the training set of TENAPP by: the model trained without contrastive learning (on the NUDT-Mobile-mini dataset), the model trained with debiased contrastive learning at a threshold of 0.9 (on the NUDT-Mobile-mini dataset), and the model trained with debiased contrastive learning at a threshold of 0.9 (train on NUDT-Mobile). Figure 3d shows the PCA result of the encoded embeddings obtained by inferring the NUDT-Mobile-mini dataset with the model trained on NUDT-Mobile.



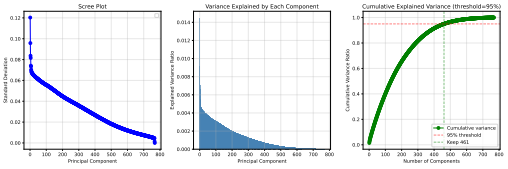
(a) Without contrastive learning (train on NUDT-Mobile-mini).



(b) With debiased contrastive learning, threshold 0.9 (train on NUDT-Mobile-mini).



(c) With debiased contrastive learning, threshold 0.9 (train on NUDT-Mobile).



(d) Inference on NUDT-Mobile-mini; the same NUDT-Mobile-trained model as in (c).

Figure 3: Principal component analysis of the embedding vectors encoded under different pretraining setups.

D.2 Training Convergence Curves

Figure 4 and Figure 5 are the pretraining loss curves (original steps and 100-step moving average), both containing the InfoNCE loss, MLM loss, MAE loss, positive-pair similarity, negative-pair similarity, and margin.

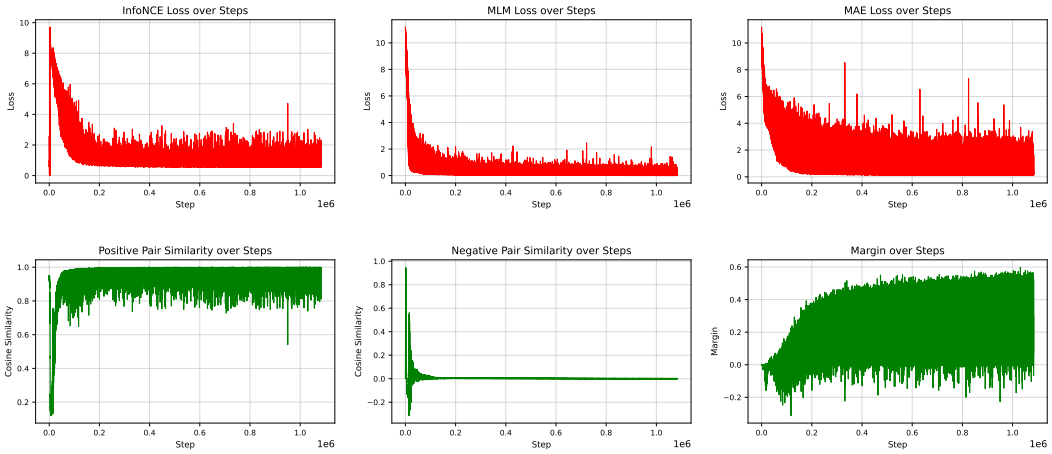


Figure 4: Pretraining loss curves (original steps).

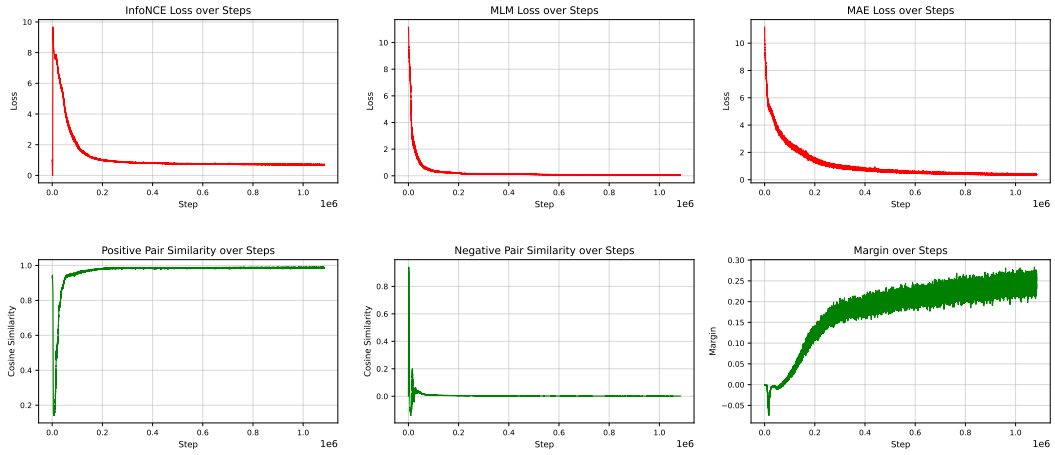


Figure 5: Pretraining loss curves (100-step moving average).

D.3 Inter-class Retrieval Margin Statistics

To further validate the class separability of the embedding space after debiased contrastive learning, we compute, for each app class in the TENAPP test set, the mean of the best intra-class retrieval similarities, the mean of the best inter-class retrieval similarities, and their difference (the average Margin). The results are shown in Table 6. The mean of the best intra-class retrieval similarities characterizes how tightly samples of the same class cluster together in the embedding space; the mean of the best inter-class retrieval similarities characterizes how confused the samples of that class are with the most similar samples of other classes. The larger their difference (Margin), the more easily that class can be distinguished from the others in the embedding space.

Table 6: Mean best intra/inter-class retrieval similarities and average Margin for each app class.

App class	Mean best intra-class sim.	Mean best inter-class sim.	Average Margin
WeChat	0.9692	0.8549	0.1143
Weibo	0.9537	0.7843	0.1695
Momo	0.9872	0.8448	0.1424
Douyin	0.9561	0.8361	0.1200
Rednote	0.9849	0.8353	0.1496
Tantan	0.9926	0.8526	0.1400
Kuaishou	0.9492	0.8361	0.1130
Bilibili	0.9863	0.7666	0.2197
QQ	0.9865	0.8398	0.1467
Soul	0.9896	0.8521	0.1376
Average	**0.9755**	**0.8303**	**0.1453**

As shown in Table 6, for all classes the mean best intra-class retrieval similarity is significantly higher than the mean best inter-class retrieval similarity, with an average Margin of about 0.1453 (the margins vary little across classes, and the smallest class margin still exceeds 0.11). From the retrieval perspective, this directly corroborates the conclusion of the contrastive-learning quality metric in Section 4.4: debiased contrastive learning keeps a certain cosine similarity gap between the positive sample and the most similar negative sample in the embedding space, thereby ensuring sufficient class separability of the retrieval results, which is also the representational geometric basis for CMTE achieving high accuracy in the fine-tuning-free retrieval scenario.

E SimilarityDPI Skill Execution Example

Session analysis report

E.1 Direct DPI Analysis

ndpiReader analysis result (global match): 1 flow

Field	Value
nDPI protocol	Unknown
Protocol stack	Unknown
TLS version	0
QUIC version	0
IP fingerprint	Unknown
Data ratio	0.128
Data ratio (description)	Mixed

Traffic statistics:

Statistic	Client→Server	Server→Client
Packet count	5	5
Byte count	380	294
Payload	160	86
Payload ratio	42.0	29.2
Mean inter-arrival time (iat)	25.0	19.0
Mean packet length	76.0	58.8
Min inter-arrival time	0	1
Max inter-arrival time	56	55
Inter-arrival time std	25.4	25.5
Min packet length	40	40
Max packet length	200	126
Packet length std	62.5	33.7
Initial window size	60	48

TCP flag statistics:

Flag	Client→Server	Server→Client
SYN	1	1
ACK	4	5
FIN	1	1
RST	0	0
PSH	1	1

DNS-related information:

- Domain corresponding to destination IP 112.45.27.46: d-pxy-access.xxpkg.com

E.2 Similarity Comparison Analysis

Comparison grouped by app class

Kuaishou

- Match count: **20** | mean similarity: **0.9919** | highest similarity: **0.9983**

Compared field	Current session	Kuaishou (Top-1, sim=0.9983)
nDPI protocol	Unknown	Unknown
Protocol stack	Unknown	Unknown
TLS version	0	0
Data ratio	0.128	0.131
CS packet count	5	5
SC packet count	5	5
CS byte count	380	380
SC byte count	294	292
CS payload ratio	42.0	42.0
SC payload ratio	29.2	28.7
Mean inter-arrival time	46.0	44.5
CS mean inter-arrival time	25.0	31.8
SC mean inter-arrival time	19.0	25.7
CS mean packet length	76.0	76.0
SC mean packet length	58.8	58.4
CS packet length std	62.5	62.5
SC packet length std	33.7	32.9
CS initial window	60	60
SC initial window	48	48
Packet length distribution	0;0;50;0;0;50;0;0;0;0;0;0;0;0;0;0;0;0;0;0;...	0;0;50;0;0;50;0;0;0;0;0;0;0;0;0;0;0;0;0;0;...

Momo

- Match count: **8** | mean similarity: **0.6667** | highest similarity: **0.7038**

Compared field	Current session	Momo (Top-1, sim=0.7038)
nDPI protocol	Unknown	HTTP
Protocol stack	Unknown	HTTP
SNI/domain	-	img.momocdn.com
TLS version	0	0
Data ratio	0.128	-0.696
CS packet count	5	5
SC packet count	5	5
CS byte count	380	386
SC byte count	294	2157
CS payload ratio	42.0	42.9
SC payload ratio	29.2	90.3
Mean inter-arrival time	46.0	45.0
CS mean inter-arrival time	25.0	37.2
SC mean inter-arrival time	19.0	15.0
CS mean packet length	76.0	77.2
SC mean packet length	58.8	431.4
CS packet length std	62.5	64.9
SC packet length std	33.7	778.8
CS initial window	60	60
SC initial window	48	48
Packet length distribution	0;0;50;0;0;50;0;0;0;0;0;0;0;0;0;0;0;0;0;0;...	0;0;0;0;0;50;0;0;0;0;0;0;0;0;0;0;0;0;0;0;...

Rednote

- Match count: **3** | mean similarity: **0.6654** | highest similarity: **0.6867**

Compared field	Current session	Rednote (Top-1, sim=0.6867)
nDPI protocol	Unknown	HTTP
Protocol stack	Unknown	HTTP
SNI/domain	-	www.xiaohongshu.com
TLS version	0	0
Data ratio	0.128	-0.108
CS packet count	5	5
SC packet count	5	5
CS byte count	380	301
SC byte count	294	374
CS payload ratio	42.0	26.8
SC payload ratio	29.2	44.3
Mean inter-arrival time	46.0	63.8
CS mean inter-arrival time	25.0	67.5
SC mean inter-arrival time	19.0	59.3
CS mean packet length	76.0	60.2
SC mean packet length	58.8	74.8
CS packet length std	62.5	31.4
SC packet length std	33.7	65.7
CS initial window	60	60
SC initial window	48	48
Packet length distribution	0;0;50;0;0;50;0;0;0;0;0;0;0;0;0;0;0;0;0;0;...	0;0;50;0;0;50;0;0;0;0;0;0;0;0;0;0;0;0;0;0;...

Final classification conclusion

Assigned class: **Kuaishou**

Confidence: high

Decision rationale:

1. **Similarity**: mean similarity 0.9919, highest 0.9983
2. **Comparison with the second candidate (Momo)**: mean similarity 0.9919 vs 0.6667 (gap 0.3252)
3. **DPI feature matching**: see the deep comparison tables above for the match/deviation of each dimensional field

F Agent and Analysis Results on Top-1 Retrieval Failures

To further evaluate the end-to-end performance and interpretability of SimCMTC in real retrieval scenarios, we conduct a per-session analysis of the **1635** session pcaps for which Top-1 retrieval failed in the TENAPP test set. The session files are randomly shuffled and renamed in numeric order to prevent class information in the filenames from biasing the agent's judgments. The experiment uses the SimilarityDPI skill to automatically generate the classification judgment and decision rationale for each session, and then introduces the ground truth for an aggregated evaluation. The result report is given below.

Classification of all masked-directory sessions vs. annotation match (analysis_by_gt)

Analysis object: all **1635** session pcaps under `mask/`.

Annotation source: `indicate.csv` (each line = `session-id.pcap original-filename`).

Prediction source: the "classification judgment" line in `result/mask/X.pcap.md`.

Note: of the annotated 1635 sessions, **all belong to the 10 app classes; there are no true open-set samples**.

F.1 Data Verification

Item	Value
Total sessions	1635
Reports generated	1635 (0 missing, 0 empty files)
Predictions parseable	1635 (0 unparsed)
Annotated app count	full coverage of 10 classes

Ground-truth distribution

App	Count
WeChat	736
Rednote	196
Weibo	177
Kuaishou	177
Soul	124
Douyin	85
Momo	66
Tantan	47
Bilibili	18
QQ	9

F.2 Overall Matching Results

- **Exact match: 1158 / 1635 = 70.83%**
- Misclassifications: 477 cases (of which **271 are predicted as "unknown class/open-set"** and 206 are predicted as wrong classes among the other 10 classes)

Confusion matrix (rows = ground-truth, columns = prediction; OP = predicted open-set)

truth\pred	WeChat	Rednote	Weibo	Kuaishou	Soul	Douyin	Momo	Tantan	Bilibili	QQ	OP	Total
WeChat(736)	545	1	4	16	8	0	0	21	2	10	**129**	736
Rednote(196)	0	152	5	1	16	0	0	0	1	0	**21**	196
Weibo(177)	0	0	**165**	1	0	3	0	2	0	0	**6**	177
Kuaishou(177)	0	1	2	**161**	1	1	1	1	2	0	**7**	177
Soul(124)	0	9	1	5	24	8	0	1	1	**27**	**48**	124
Douyin(85)	0	5	3	0	3	**52**	0	0	0	0	**22**	85
Momo(66)	0	2	1	5	1	0	30	4	0	0	**23**	66
Tantan(47)	0	5	0	1	3	0	4	9	0	**14**	**11**	47
Bilibili(18)	0	0	0	0	0	0	0	0	**15**	2	**1**	18
QQ(9)	1	0	0	0	0	0	0	0	0	5	**3**	9

F.3 Per-class Quality (Recall / Precision)

Class	Correct	Recall	Precision
WeChat	545/736	**74.0%**	**99.8%**
Rednote	152/196	**77.6%**	86.9%
Weibo	165/177	**93.2%**	91.2%
Kuaishou	161/177	**91.0%**	84.7%
Soul	24/124	**19.4%**	42.9%
Douyin	52/85	**61.2%**	81.2%
Momo	30/66	**45.5%**	85.7%
Tantan	9/47	**19.1%**	23.7%
Bilibili	15/18	**83.3%**	71.4%
QQ	5/9	**55.6%**	**8.6%**

Key conclusions

1. **Strengths**: WeChat (although 129 open-set cases lower its recall, **almost every WeChat prediction is correct**, with 99.8% precision), Weibo, Kuaishou, Rednote, and Bilibili achieve high recognition quality.
2. **Weaknesses**:

- **Soul / Tantan have the lowest recall (~19%)**, with many cases misclassified as **QQ** (Soul→QQ 27 cases, Tantan→QQ 14 cases). All three are dating/social-networking apps that frequently carry Tencent-ecosystem SDKs (the QQ system) and similar session patterns, which are hard to distinguish with the current features.
- **QQ precision is only 8.6%** (only 5 of the 9 true QQ are correct, yet many Soul/Tantan/WeChat sessions are predicted as QQ).
- **Douyin / Momo recall is low** (61%/45%), mainly leaking to open-set (third-party ByteDance infrastructure, unidentifiable sessions).
- **Largest single source of missed detections**: **129 WeChat cases are judged as open-set** (see Section F.4), accounting for nearly half of all 271 open-set cases.

F.4 Summary of Sessions Predicted as "Open-Set" (271 cases in total)

Because there is no true open set in the annotations, these 271 cases are **all "false-open-set" misses of known apps**. Each case's open-set rationale is stated in its own report; organized by rationale, they fall into 7 categories.

Categorized by open-set rationale

The 271 cases fall into seven categories by their stated rationale:

- **SYN→RST connection failure / empty-payload flows (92 cases)**: the session repeatedly performs SYN→RST, with zero goodput and no SNI/UA; nDPI carries no identifier.
- **Baidu ecosystem / map SDK (35 cases)**: third-party location/map SDKs such as `api.map.baidu.com` and `cnloc.map.baidu.com`, which are not among the ten classes.
- **Third-party live-streaming / lending / other platforms (37 cases)**: domains such as `tlivesource.com`, `zztfly.com`, `ihani.tv`, and `g-lending.com`, together with lending, Guangdiantong, and ad platforms.
- **Third-party push / statistics / risk-control SDKs (33 cases)**: Getui (`getui`), JPush (`jpush`), MiaoZhen (`miaozhen`), Shumei (`fengkongcloud`), and device fingerprinting.
- **General CDN / static / ByteDance infrastructure (33 cases)**: generic static/CDN domains such as `bytemastatic.com` and ByteDance internal infrastructure.
- **Self-media / app-less HTTP downloads (29 cases)**: bare IP or Unknown protocol, large-packet downloads with no SNI and no UA, carrying no identifier and no reliable same-source embedding.
- **Other / combined (12 cases)**: a combination of multiple factors like the highest similarity below 0.5.

DISTRIBUTION BY THE MISJUDGED TRUE APP

True App	Count judged open-set
WeChat	**129**
Soul	48
Momo	23
Douyin	22
Rednote	21
Tantan	11
Kuaishou	7
Weibo	6
QQ	3
Bilibili	1

KEY INTERPRETATION

- These open-set judgments are **mostly reasonable at the session-content level**: the session indeed carries no "content signal belonging to a specific one of the 10 apps" (pure connection failure, pure SDK request, bare download, etc.), and the skill's "open-set" rule (judge open-set when there is no credible identifier or the similarity is insufficient) is correct.
- However, **at the annotation level**, a session initiated by an app (e.g., a push-SDK request of WeChat, a map call of Soul) still belongs to that app, thus statistically forming a "false-open-set" miss.
- WeChat contributes the most (129 cases), mainly from: WeChat SDK / Tencent-ecosystem third-party services (Tencent Maps `lhp.map.qq.com`, Tencent push / browser X5 kernel, etc., which are classified as open-set but are actually initiated by WeChat), as well as SYN→RST connection failures.

F.5 MAIN SCENARIOS OF MISCLASSIFICATION TO OTHER APPS (NON-OPEN-SET, 206 CASES)

The main misclassification patterns are as follows:

- **Soul → QQ (27 cases)**: Soul sessions that really carry QQ-system SDKs; their embeddings/DPI land on QQ.
- **Tantan → QQ (14 cases)**: Tantan sessions recognized as the QQ system.
- **WeChat → Tantan / Kuaishou (21 / 16 cases)**: WeChat sessions whose form resembles Tantan/Kuaishou.
- **Rednote → Soul (16 cases)**: Rednote sessions judged as Soul.
- **Momo → Kuaishou / Tantan (5 / 4 cases)**: Momo sessions falling into Kuaishou/Tantan.
- **Soul → Rednote / Douyin (9 / 8 cases)**: social/video form confusion.

These misclassifications mostly occur on sessions **without explicit SNI/UA and relying only on vector similarity**: short HTTP/TLS flows of similar form have close vector distances across different social/content apps and are therefore easily confused.