Rosetta: Enabling Robust TLS Encrypted Traffic Classification in Diverse Network Environments with TCP-Aware Traffic Augmentation

Renjie Xie and Jiahao Cao, *Tsinghua University;* Enhuan Dong and Mingwei Xu, *Tsinghua University and Quan Cheng Laboratory;* Kun Sun, *George Mason University;* Qi Li and Licheng Shen, *Tsinghua University;* Menghao Zhang, *Tsinghua University and Kuaishou Technology*

https://www.usenix.org/conference/usenixsecurity23/presentation/xie

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

Rosetta: Enabling Robust TLS Encrypted Traffic Classification in Diverse Network Environments with TCP-Aware Traffic Augmentation

Renjie Xie[1,3,*], Jiahao Cao[1,2,3,*], Enhuan Dong[1,3,5,*], Mingwei Xu[1,2,5], Kun Sun[4]
Qi Li[1], Licheng Shen[2] and Menghao Zhang[2,6]

[1]*Institute for Network Sciences and Cyberspace, Tsinghua University*
[2]*Department of Computer Science and Technology, Tsinghua University*
[3]*Beijing National Research Center for Information Science and Technology, Tsinghua University*
[4]*Department of Information Sciences and Technology, George Mason University*
[5]*Quan Cheng Laboratory,* [6]*Kuaishou Technology*

Abstract

As the majority of Internet traffic is encrypted by the Transport Layer Security (TLS) protocol, recent advances leverage Deep Learning (DL) models to conduct encrypted traffic classification by automatically extracting complicated and informative features from the packet length sequences of TLS flows. Though existing DL models have reported to achieve excellent classification results on encrypted traffic, we conduct a comprehensive study to show that they all have significant performance degradation in real diverse network environments. After systematically studying the reasons, we discover the packet length sequences of flows may change dramatically due to various TCP mechanisms for reliable transmission in varying network environments. Thereafter, we propose Rosetta to enable robust TLS encrypted traffic classification for existing DL models. It leverages TCP-aware traffic augmentation mechanisms and self-supervised learning to understand implict TCP semantics, and hence extracts robust features of TLS flows. Extensive experiments show that Rosetta can significantly improve the classification performance of existing DL models on TLS traffic in diverse network environments.

1 Introduction

Network traffic classification aims to organize various traffic into different categories, which is fundamental and vital for network management and security. A number of network security tasks have been built on top of it, such as application identification [53, 56, 65], website fingerprinting [46, 49, 51, 52], malicious flow detection [33, 37, 38], and user profiling [16, 23]. With the fast-growing need of user privacy protection and the wide usage of the Transport Layer Security (TLS) protocol, a majority of the Internet traffic has been encrypted [12]. Traditional rule-based methods that examines packet payloads are becoming increasingly ineffective in classifying encrypted network traffic [6, 34].

Recent advances [16, 33, 38, 49, 51, 52] are leveraging deep learning (DL) techniques to conduct generic traffic classification. Particularly, as the packet payloads are converted into pseudorandom values after TLS encryption [36], a number of studies [7, 35, 46, 49–51] design various deep learning models to automatically extract complicated and high-level features from packet length sequences, which possess rich and discriminating implicit information of the encrypted flows. Besides, it is convenient and low-cost to measure and derive packet length sequences in real-world large-scale networks, even supporting real-time traffic classification tasks [5, 6].

Though these DL models have been reported to achieve excellent classification results on encrypted traffic [3, 7, 13, 14, 41, 49], e.g., 98% classification accuracy [49], the performance of these models for various traffic classification tasks in the real-world diverse network environments is still not clear. We should note that when they are deployed in a real network for TLS traffic classification, they will face diverse network environments that are time-varying and unpredictable. For example, the packet loss rate and network delay may suddenly arise due to the burst of network traffic [48, 63]. Actually, the environment of one network can be changed complexly due to the joint effect of multiple factors, such as traffic burst [48, 63], traffic engineering [22, 54], partial network failures [2, 64], and network updates [11, 45].

In this paper, we conduct a systematic study to check if existing deep learning models can effectively classify TLS encrypted traffic in diverse network environments. We study six different DL models including Deep Fingerprinting (DF) [51], FS-Net [35], Transformer [57], SDAE [4, 7, 46, 51], CNN [7, 46, 49, 51, 59], and LSTM [7, 46, 51, 59] that rely on packet length sequences to classify encrypted traffic. We conduct experiments not only with the replayed traffic from two typical TLS traffic datasets [19, 39] in diverse network environments, but also with real TLS traffic that is generated by visiting popular websites and running online network applications in diverse Internet environments. Our experiments confirm that all these DL models can achieve excellent results with the offline TLS traffic dataset for various clas-

*These authors contributed equally to this work.

sification tasks, including website fingerprinting, malicious flow identification, VPN traffic identification, and application fingerprinting. However, the performance of all models drops remarkably when they are tested in different network environments, e.g., about 53% accuracy drop at worst.

We find that the remarkable performance degradation results from the dramatic change of packet length sequences of the same flow in different network environments. For example, a TLS encrypted flow with the packet length sequence $[q_1, q_2, q_3, q_4]$ may change to $[q_3, q_2, q_1, q_4]$ due to high packet loss in another network environment. However, existing DL models fail to understand that the two different packet length sequences in different network environments actually originate from the same flow. Furthermore, we notice that the changes of packet length sequences follow the TCP specifications in different network environments, since TLS connections are built on the TCP protocol. Consequently, different TCP mechanisms ensuring reliable transmission in diverse network environments cause three major changes of packet length sequences, i.e., packet subsequence shift, packet subsequence duplication, and packet size variation. Thus, if a model can be aware of these regular packet sequence changes with TCP semantics, robust TLS encrypted traffic classification in diverse network environments may be achieved.

To this end, we develop Rosetta that is capable of enhancing robust TLS encrypted traffic classification for existing deep learning models. The main idea is to learn implicit TCP semantics from carefully crafted traffic and generate effective feature vectors that represent robust features of TLS flows in diverse network environments. Hence, existing deep learning models can leverage these feature vectors to achieve robust TLS encrypted traffic classification. Rosetta consists of two modules: *TCP-aware traffic augmentation* and *traffic invariant extractor*. We develop TCP-aware traffic augmentation algorithms based on a thorough understanding of TCP mechanisms that may affect packet length sequences of flows. Hence, we can generate massive flows that reflect how TLS flows may change in various network environments. The traffic invariant extractor applies self-supervised learning to extract robust features by projecting flow variants into a proper hidden space, reducing the distance among feature vectors of flow variants from the same flow. Consequently, flow variants coming from the same flow will have similar feature vectors.

We conduct extensive experiments to evaluate the effectiveness of Rosetta. The results show that Rosetta significantly improves the performance of existing deep learning models on traffic classification in diverse network environments with both replayed and real TLS flows. We further evaluate its classification robustness under different packet loss rates and different delays. Without enabling Rosetta, the classification accuracy of existing models drops remarkably when packet loss rates and delays increase. For example, the accuracy drops from 99% to 55% when the delay is increased from 0 to 50 ms. When Rosetta is enabled, the accuracy can always maintain above 86%. Moreover, we compare our TCP-aware traffic augmentation algorithms with classical data augmentation methods, including Random Mask (RM) [17] and Random Swap (RS) [60] that have been widely used in the domains of Natural Language Processing (NLP) and Computer Version (CV). With RM and RS, the average F1-score is less than 47% in six different network environments. With our TCP-aware traffic augmentation, the average F1-score is 87%. The results demonstrate that TCP-aware traffic augmentation is more effective on extracting robust features of TLS flows in different network environments.

In summary, we make the following contributions:

- We conduct comprehensive experiments to show that mainstream DL models cannot robustly classify TLS encrypted traffic in different network environments.
- We propose Rosetta that can enable robust TLS encrypted traffic classification for the DL models based on TCP-aware traffic augmentation mechanisms and self-supervised learning.
- We conduct extensive experiments to demonstrate the effectiveness of Rosetta on improving the encrypted traffic classification performance of existing DL models in various network environments.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 evaluates the mainstream deep learning models in diverse network environments and analyzes the performance degradation of the models. Section 4 presents the design of Rosetta, including five TCP-aware traffic augmentation algorithms and the traffic invariant extractor. Section 5 evaluates the effectiveness of Rosetta. Section 6 discusses the limitations of Rosetta. Section 7 concludes the paper. For the research community to utilize our tool, we release the source code in Github: `https://github.com/sunskyXX/Rosetta.git`.

2 Related Work

Encrypted Traffic Classification. Though there are studies [3, 13, 14, 20, 38, 41] relying on experts to extract and select effective features of encrypted flows, recent advances [16, 27, 33, 34, 36, 38, 49, 51, 52, 62] focuses more on automatic feature extraction and encrypted traffic classification through deep learning from raw traffic inputs. Particularly, as packet length sequences of encrypted flows poss rich and discriminating implicit information, various deep learning models are presented to classify encrypted traffic from packet length sequences. Cao et al. [7] employ CNN, LSTM, and SDAE to fingerprint applications of Software Defined Networking (SDN) from packet length sequences of TLS encrypted control traffic. Shen et al. [50] accurately fingerprint decentralized applications using graph neural networks based

Table 1: Different Network Environments for Relayed Traffic

Network Type	Env. ID	Sender Loc.	Receiver Loc.	Access mode
Wired	θ_0	Local LAN	Local LAN	Ethernet
	θ_1	China	China	
	θ_2	Korea	China	
	θ_3	USA	China	
Wireless	θ_4	China	China	Wi-Fi
	θ_5	China	China	4G LTE
	θ_6	China	China	3G WCDMA

on packet length sequences of encrypted flows. Many studies [46, 49, 51] leverage deep learning to achieve automatic website fingerprinting from packet length sequences of encrypted Tor traffic. Liu et al. [35] present Flow Sequence Networks (FS-Net) to learn representations from packet length sequences and achieve accurate encrypted traffic classification. They can achieve high accuracy on encrypted traffic classification with traffic datasets tested in an offline environment. Cherubin et al. [10] further evaluate website fingerprinting in a real-word online network environment. However, it does not systematically evaluate how the performance of website fingerprinting changes in diverse network environments. In contrast, our work conducts the systematical study to evaluate the performance of the existing models in diverse network environments with various TLS traffic classification tasks, including website fingerprinting, malicious flow identification, VPN traffic identification, and application fingerprinting. We also present Rosetta to enable robust encrypted traffic classification in diverse network environments.

Data Augmentation Techniques. Data augmentation has been widely used in the domains of Computer Vision (CV) and Natural language processing (NLP), which can increase training data diversity without collecting more data. There are a number of image augmentation techniques in CV, including geometric transformations [40], kernel filters [29], mixing images [28], random erasing [66], and feature space augmentation [18]. Due to the unique features of languages, NLP applies predetermined transform rules [25, 60] to augment data, such as token-level random perturbations [60]. Besides, researchers [17, 21, 31, 47] train different models to augment NLP data according to downstream tasks. For example, Back-translation [47] generates data by translating a sequence into another language and then back into the original language. Recent advances [15, 30, 42] apply Generative Adversarial Networks (GAN) to conduct data augmentations. Our work differs from them as we focus on how to augment TLS traffic with TCP semantics in diverse network environments.

3 Measurement Study

In this section, we perform a systematic study on the performance of six representative deep learning models on TLS encrypted traffic classification when being trained in one network environment and then deployed in different network environments. We first conduct experiments with replayed TLS flows from two typical traffic datasets that have been widely used for traffic classification based on deep learning. We further conduct experiments with real TLS flows that are generated by visiting popular websites and running real applications. All experiments demonstrate the similar performance degradation of existing traffic classification methods when training and deploying in different network environments. Finally, we analyze the reasons of the performance degradation.

3.1 Experiments with Relayed TLS Traffic

3.1.1 Experiment Setup

Diverse Network Environment Construction. We leverage a sender host and a receiver host on the Internet to replay TLS encrypted flows between them. By changing the locations of the two hosts, the flows can experience real diverse network environments. This is because the sender and the receiver are in different networks and the routing paths of the flows are also different. Therefore, we create four diverse wired network environments by changing the sender location to four different countries, which is shown in Table 1. Besides, we create three different wireless access network environments by changing the access mode of the sender to Wi-Fi, 4G LTE, and 3G WCDMA. For simplicity, we name the above different network environments as θ_0 to θ_6, respectively.

Replayed TLS Traffic Collection. Our experiments leverage two typical TLS encrypted traffic datasets, i.e., CIRA-CIC-DoHBrw-2020 [39] and ISCX-VPN [19]. They have been widely used for traffic classification based on deep learning [8, 34, 36]. CIRA-CIC-DoHBrw-2020 has 249,750 malicious flows and 917,300 benign flows. ISCX-VPN has 16,007 VPN-encapsulated flows and 22,716 non-VPN encapsulated flows. To better simulate application behaviors and make flows experience diverse network environments, we extract the payloads and timestamps of each packet from encrypted flows. As the dataset contains retransmitted packets, we reorder the packet according to their timestamps and remove the duplicate packets. Based on the timestamps, we send the payloads via TCP sockets established in different network environments θ_0 to θ_6, respectively. Here, we replace the original source and destination IP addresses with the sender and receiver IP addresses in our experiments.

To replay the traffic in wireless network environments, we use a WIFI router that connects to the Internet, and the sender host further connects to the WIFI router. To replay the traffic in the access mode of 4G LTE or 3G WCDMA in the wireless network environment, we mount a cellphone supporting 4G LTE and 3G WCDMA to the sender host via USB, and we configure the cellphone to act as a network adapter for the sender host. Thus, the host can access the Internet through the

Table 2: Results on Classifying Replayed TLS Flows from CIRA-CIC-DoHBrw-2020 in Different Network Environments.

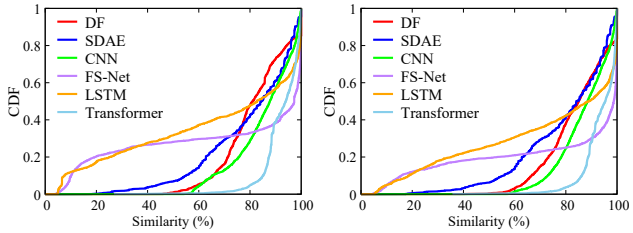
Model	Different Wired Network Environments								Different Wireless Access Network Environments					
	θ_0		θ_1		θ_2		θ_3		θ_4		θ_5		θ_6	
	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
CNN	99.89%	99.84%	98.21%	98.20%	53.16%	34.91%	57.04%	36.32%	87.47%	87.03%	74.42%	71.52%	53.26%	34.96%
SDAE	95.47%	95.46%	91.47%	91.47%	56.21%	43.40%	55.75%	36.04%	88.11%	88.03%	82.11%	81.42%	55.16%	41.73%
LSTM	95.26%	95.25%	87.68%	87.47%	53.05%	35.07%	57.04%	36.57%	82.00%	81.19%	70.84%	67.34%	53.58%	36.08%
DF	99.89%	99.84%	98.42%	98.41%	53.26%	34.75%	58.03%	36.72%	88.00%	87.57%	74.95%	72.17%	53.37%	35.00%
FS-Net	92.11%	92.10%	90.74%	90.71%	61.16%	52.11%	58.10%	39.66%	88.84%	88.76%	83.68%	83.30%	56.84%	44.50%
Transformer	99.56%	99.36%	98.28%	96.00%	62.22%	54.12%	57.04%	42.00%	93.74%	91.35%	85.62%	83.12%	54.27%	47.57%
On Average	**97.03%**	**96.98%**	**94.13%**	**93.71%**	**56.51%**	**42.39%**	**57.17%**	**37.89%**	**88.03%**	**87.32%**	**78.60%**	**76.48%**	**54.41%**	**39.97%**

cellular networks that the cellphone provides. In each network environment, we capture the packet length sequences of the replayed flows through the receiver host with `tcpdump` and feed them into deep learning models for classification.

Model Training and Testing. We consider six representative deep learning models in our experiments, i.e., Denoising Autoencoder (SDAE) [58], Convolutional Neural Network (CNN) [32], Long-Short Term Memory (LSTM) [26], Deep Fingerprinting (DF) [51], FS-Net [35], and Transformer [57]. The first three models achieve good results in website fingerprinting based on encrypted traffic [46]. DF improves the performance of website fingerprinting in the situation where the traffic is mixed with artificially added noise. FS-Net learns representative features from encrypted flows and enforces accurate traffic classification. Transformer has been one of the state-of-the-art methods in various classification tasks. To explore the performance of the encrypted traffic classification models in diverse network environments, we first train and test them with packet length sequences of flows captured in the network environment θ_0. In the training network environment θ_0, the ratio on the number of flows in the training set, validation set, and testing set is 6:2:2. Next, we test all the trained models with packet length sequences of flows captured in the other network environments, i.e., θ_1 to θ_6. Similarly, we train the classification models in the network environments θ_1 to θ_6, respectively, and test against the other six different network environments. Here, following the general settings of existing methods [51, 52], we fix the length of the input sequence, i.e., any sequence longer than 100 will be truncated to 100, and any sequence shorter than 100 is padded with zeros. Besides, as different flows are naturally unbalanced [34, 35], we use the original ratio of different flows to train and test the existing models on TLS traffic classification in the real-world network environment. This is also the typical way where most of the existing studies apply [34, 35].

3.1.2 Experimental Results

Degraded Classification Performance. We apply two metrics, i.e., accuracy (AC) and F1-Score (F1), to evaluate the traffic classification results. As shown in Table 2, all six deep learning models achieve more than 92% accuracy and F1-Score for the replayed CIRA-CIC-DoHBrw-2020 flows in the network environment θ_0. Note that all the models are trained in the network environment θ_0. However, when applying these well-trained models in other network environments (i.e., θ_1 to θ_6), we observe a significant drop in their performance[1]. For example, the F1-Score of CNN drops from 99.84% in the network environment θ_0 to 34.96% in the network environment θ_6. We can see that the average accuracy of all the models decreases by 2.9% at least to 42.62% at most in the network environments θ_1 to θ_6, and the average F1-Score decreases by 3.27% at least and 58.09% at most. The experimental results with the replayed flows from ISCX-VPN demonstrate similar results. Please refer to Table 12 in Appendix C for detailed results. It is obvious that different network environments can significantly affect the traffic classification results of existing deep models. The results demonstrate that existing deep learning models fail to enable robust TLS encrypted traffic classification in diverse network environments.

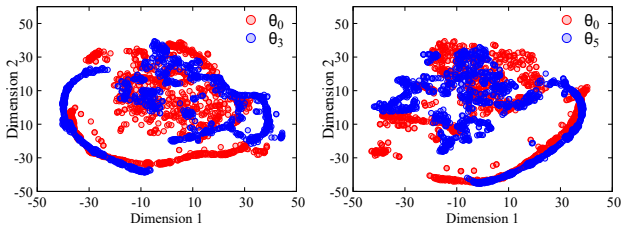


(a) Similarity between θ_0 and θ_3. (b) Similarity between θ_0 and θ_5.

Figure 1: CDF of the cosine similarity for all pairs of feature vectors between two different network environments.

Varying Feature Vectors in Different Network Environments. The layer before the last output layer of a deep learning model will generate a feature vector that contains the hidden abstract features of the input sample. We extract the feature vectors of the replayed flow in each of the seven different network environments to see if the existing deep learning models

[1] For simplicity, we only show the results using θ_0 as the training network environment. The results are similar when using other environments as the training environment.



(a) Distribution between θ_0 and θ_3. (b) Distribution between θ_0 and θ_5.

Figure 2: Distribution of feature vectors between two different network environments.

can extract robust hidden features. Specifically, we collect a pair of feature vectors for the replayed flows in two different network environments and calculate their cosine similarity. Figure 1 shows the Cumulative Distribution Function (CDF) of the cosine similarity for all pairs of feature vectors between two different network environments. Figure 1a demonstrates that more than 30% pairs of feature vectors between the two network environments θ_0 and θ_3 have less than 70% cosine similarity for most models. There are a few pairs of feature vectors that have less than 10% cosine similarity for the FS-Net model. It shows that the extracted features for a replayed flow in different network environments are remarkably biased. Figure 1b demonstrates similar results for the feature vectors between network environments θ_0 and θ_5.

To better illustrate how the feature vectors change in different network environments, we apply the t-SNE [55] method to project high-dimensional feature vectors extracted by the DF model into 2D vectors. Hence, we can draw all the vectors in 2D space, which is shown in Figure 2. From the results, we can see that the distribution of feature vectors changes significantly when the network environment is changed. These results show that existing deep learning models fail to extract a fixed set of robust hidden features for the flows replayed and collected in different network environments.

3.2 Experiments with Real TLS Traffic

3.2.1 Experiment Setup

Diverse Network Environment Construction. We rent hosts in different countries as clients to visit popular websites with the Selenium web driver [43] driving the Firefox browser. By changing the locations of the clients, the routing paths between the clients and website servers are different. Thus, there are three different wired network environments, which is shown in Table 3. Besides, we make flows experience three different wireless access network environments by changing the access mode of the client to Wi-Fi, 4G LTE, and 3G WCDMA, respectively. For simplicity, we name the above different network environments as τ_1 to τ_6, respectively. Furthermore, we run real applications on these clients in diverse network environments to collect real application TLS traffic in a similar way where we collect the website TLS traffic.

Table 3: Different Network Environments for Real Traffic

Network Type	Env. ID	Client Loc.	Access mode
Wired	τ_1	China	Ethernet
	τ_2	Korea	
	τ_3	USA	
Wireless	τ_4	China	Wi-Fi
	τ_5	China	4G LTE
	τ_6	China	3G WCDMA

Real TLS Traffic Collection. We collect about 1.8 million website TLS flows by visiting popular websites for 14 days, including Alipay, Apple, Baidu, iCloud, JD, Kaipanla, NeCmusic, QQ, Sogou, Weibo, Youdao and Zhihu. To make the flows experience diverse network environments, we leverage the clients in network environments τ_1 to τ_6 to generate traffic by visiting the above websites. Furthermore, we collect 292,523 application TLS flows by running popular applications in the above six diverse network environments, including TencentVideo, YoudaoNote, QQmusic, NeCmusic, Youku, and BaiduNetdisk.

Model Training and Testing. To explore the performance of the encrypted traffic classification models for real website TLS traffic and application TLS traffic in diverse network environments, we apply SDAE, CNN, LSTM, DF, FS-Net and Transformer to classify the flows collected from τ_1 to τ_6. The classification task for the models is to classify real TLS flows that are generated by different websites/applications. Given a TLS flow, the models identify which website/application generates it. We first train and test them with packet length sequences of flows captured in the network environment τ_1. Next, we test all the trained models with packet length sequences of flows captured in the other network environments, i.e., τ_2 to τ_6.

3.2.2 Experimental Results

Results with Real Website TLS traffic. As shown in Table 4, the six deep learning models achieve more than 83% accuracy and F1-Score on average for the website TLS flows in the network environment τ_1. Note that all the models are trained in network environment τ_1. However, when applying these well-trained models in other network environments, we observe a significant performance drop. For example, the F1-Score of DF drops from 91.15% in τ_1 to 66.91% in τ_5. We see that the average accuracy of all the models decreases by 8.57% at least to 19.09% at most in τ_2 to τ_6. The average F1-Score decreases by 8.18% at least to 19.82% at most.

Results with Real Application TLS Traffic. As shown in Table 5, the six deep learning models achieve about 80% accuracy and 76% F1-Score for application TLS flows in τ_1 on average. Note that all the models are trained in τ_1. However, when applying these well-trained models to classify application flows in the other network environments, we observe the

Table 4: Results on Classifying Real Website TLS Flows in Different Network Environments.

Model	Different Wired Network Environments						Different Wireless Access Network Environments					
	τ_1		τ_2		τ_3		τ_4		τ_5		τ_6	
	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
CNN	89.55%	89.28%	81.48%	80.88%	57.73%	52.29%	72.51%	68.51%	67.16%	60.15%	70.63%	68.73%
SDAE	82.37%	79.79%	78.13%	74.79%	70.04%	68.80%	68.04%	67.98%	64.57%	64.20%	69.94%	64.01%
LSTM	81.85%	77.39%	76.72%	74.08%	62.71%	57.26%	60.89%	60.04%	66.93%	63.60%	66.41%	61.67%
DF	91.27%	91.15%	83.95%	80.58%	83.59%	83.50%	79.90%	75.00%	70.67%	66.91%	73.03%	70.17%
FS-Net	85.81%	81.42%	73.02%	72.20%	64.42%	61.97%	70.14%	68.39%	64.84%	65.42%	67.65%	66.48%
Transformer	84.85%	82.13%	70.97%	69.57%	62.66%	58.46%	63.71%	62.14%	78.98%	75.38%	61.37%	59.74%
On Average	**85.95%**	**83.53%**	**77.38%**	**75.35%**	**66.86%**	**63.71%**	**69.20%**	**67.01%**	**68.86%**	**65.94%**	**68.17%**	**65.13%**

Table 5: Results on Classifying Real Application TLS Flows in Different Network Environments.

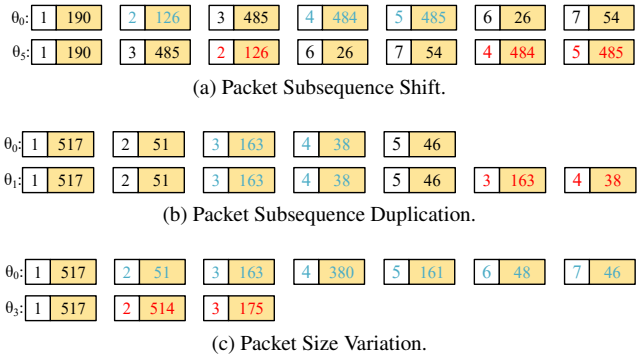
Model	Different Wired Network Environments						Different Wireless Access Network Environments					
	τ_1		τ_2		τ_3		τ_4		τ_5		τ_6	
	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
CNN	95.24%	95.07%	73.82%	71.39%	85.62%	85.14%	91.83%	91.53%	96.23%	95.91%	95.08%	94.57%
SDAE	81.67%	81.38%	53.36%	53.00%	58.75%	58.54%	76.52%	76.51%	72.71%	72.66%	59.19%	59.11%
LSTM	68.07%	57.13%	67.66%	41.58%	52.83%	38.42%	63.84%	55.35%	69.36%	57.58%	72.07%	61.04%
DF	98.11%	98.06%	83.47%	79.30%	69.95%	69.90%	92.96%	92.74%	98.69%	98.60%	98.99%	98.91%
FS-Net	69.62%	60.62%	70.79%	41.54%	56.36%	46.52%	59.63%	45.71%	68.25%	54.14%	71.06%	58.31%
Transformer	92.04%	91.35%	96.95%	96.46%	92.19%	92.11%	90.22%	90.03%	93.44%	92.46%	92.43%	92.33%
On Average	**84.13%**	**80.60%**	**74.34%**	**63.88%**	**69.28%**	**65.11%**	**79.17%**	**75.31%**	**83.11%**	**78.56%**	**81.47%**	**77.38%**

average performance of the models drops. Particularly, the F1-Score of DF drops from 98.06% in τ_1 to 69.90% in τ_3. The results show that existing deep learning models fail to enable robust TLS encrypted traffic classification in diverse network environments.

3.3 Understanding Performance Degradation

After comparing the packet length sequences of the flows collected in diverse network environments, we observe that the dramatic change on the packet length sequences exists in most flows when the network environment changes. The reason for the poor performance is that the models are not aware of dramatic change caused by network environments, even though the TLS flows in diverse network environments are from the same website or the same application.

Our further investigation reveals that the changes of packet length sequences in different network environments are mainly due to three phenomenons, namely, *packet subsequence shift*, *packet subsequence duplication*, and *packet size variation*. Figure 3 shows examples of the three phenomenons. The case of packet subsequence shift is shown in Figure 3a, the second, fourth, and fifth element of sequence in θ_0 is shifted into the third, sixth, and seventh position of sequence in θ_5. Figure 3b shows the case of packet subsequence duplication, where the the sixth, seventh, and eighth elements of sequence in θ_1 is the duplication of the second, third, and fourth elements of sequence in θ_0. Figure 3c shows the case



(a) Packet Subsequence Shift.

(b) Packet Subsequence Duplication.

(c) Packet Size Variation.

Figure 3: Three phenomenons that cause the changes of packet length sequences in diverse network environments.

of packet size variation. The second element of the sequence in θ_3 is equal to the sum of the second, third, and fourth elements of the sequence in θ_0 minus twice the header length. Also, the third element of sequence in θ_3 is equal to the sum of the fifth, sixth, and seventh elements of sequence in θ_0 minus twice the header length.

As the TLS connections are established on the TCP protocol, TLS traffic follows the specification of TCP protocol. Therefore, we are able to explain the root causes of the three phenomenons from the perspective of the TCP protocol. Due to space limitation, please refer to our additional technical report [1] for our detailed analysis. We discover that different TCP mechanisms ensuring reliable transmission in di-

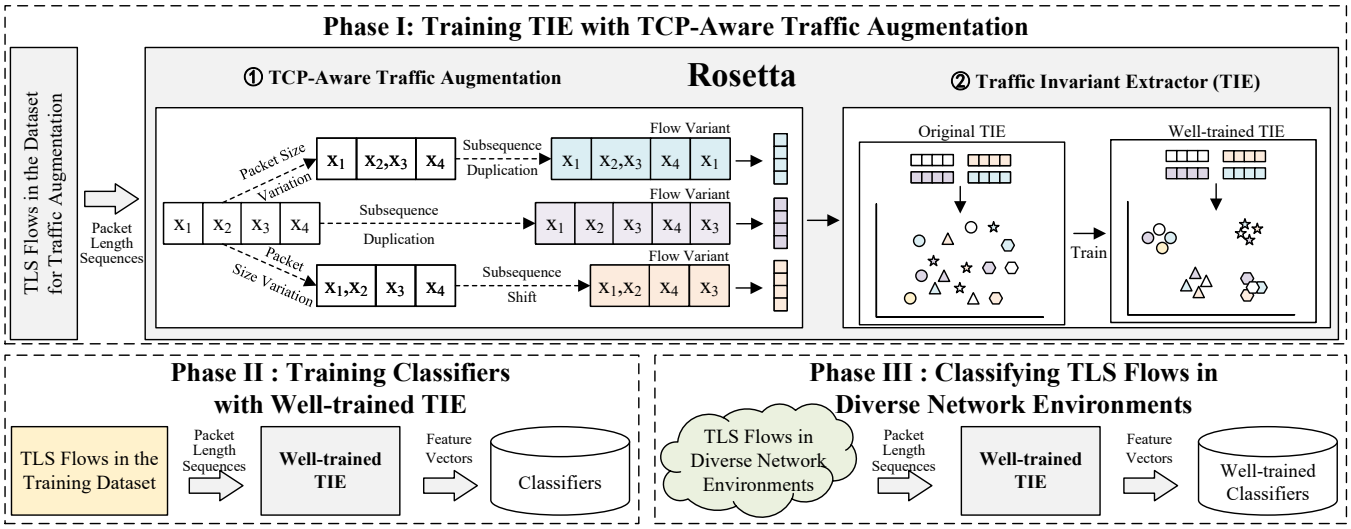


Figure 4: The workflow of Rosetta-augmented encrypted network traffic classification: (1) In Phase I, Rosetta leverages TCP-aware traffic augmentation to generate massive flow variants for packet length sequences of the TLS flows from a traffic dataset. These flow variants are used to train TIE for robust feature extraction in a self-supervised manner; (2) In Phase II, the well-trained TIE extracts the robust feature vectors from packet length sequences of TLS flows in the training dataset. The feature vectors and their labels are used to train deep learning classifiers in a supervised manner; (3) In Phase III, the well-trained classifiers classify the TLS flows according to their feature vectors that are extracted by the well-trained TIE.

verse network environments cause the three phenomenons. TCP retransmission timeout (RTO) and fast retransmit mechanisms can both cause packet subsequence shift and packet subsequence duplication when there is packet loss in diverse network environments. The Nagle algorithm and the maximum transmission unit (MTU) impacts packet size of TLS flows when there are high delays or different MTUs in network environments. However, we find that the changes of packet length sequences always follow the TCP specifications no matter how network environments change. If we can be aware of these regular packet sequence changes with TCP semantics, robust TLS traffic classification in diverse network environments may be achieved.

4 System Design of Rosetta

4.1 Threat Model

We consider the scenario where there are various TLS traffic classification tasks. These deep learning models automatically extract complicated and high-level features from packet length sequences of TLS flows [7,35,46,49–51]. We consider that they are deployed in a real-world network for TLS traffic classification. As the network environment of the real-world network is time-varying and unpredictable, they face the diverse network environments. Various factors can change the environment of one network, such as traffic burst [48, 63], traffic engineering [22, 54], partial network failure [2, 64], and network update [11, 45].

Rosetta aims to improve the traffic classification robustness of existing deep learning models in the real-world diverse network environments. We do not require Rosetta knowing the prior knowledge of the network and its network environments. Actually, it is almost impossible to know the time-varying and unpredictable network environments. We do not require a substantial change for existing models to be compatible with Rosetta. We require changing the dimension of the input layer of existing models to the dimension of the robust features extracted by Rosetta.

4.2 Overview

We propose Rosetta to extract robust features from packet length sequences of TLS encrypted traffic. Rosetta consists of two modules: TCP-aware traffic augmentation and traffic invariant extractor (TIE). The first module generates a number of variants for each flow with TCP-aware traffic augmentation algorithms. Hence, we can get a number of new flows that simulate how the original flows change in different network environments. The second module extracts similar and robust features from packet length sequences of flows. Hence, it can understand the potential flow changes in different network environments and whether the flow variants belong to the same flow. To achieve this goal, it applies self-supervised learning to project flow variants into a proper hidden space as different feature vectors. It reduces the distance among feature vectors of flow variants coming from the same flow.

Rosetta aims to improve the performance of mainstream DL models on traffic classification in diverse network environments. Figure 4 shows the workflow after integrating

Rosetta with mainstream DL models. Rosetta does not require substantial changes for the network structures of the existing DL models. By changing the dimension of the input layer to the dimension of the feature vector, existing DL models are completely compatible with Rosetta. In the following subsections, we will detail the two core modules of Rosetta, i.e., TCP-aware traffic augmentation and traffic invariant extractor.

4.3 TCP-Aware Traffic Augmentation

As we mentioned in Section 3.3, the changes of packet length sequences of flows are mainly due to three phenomenons, i.e., packet subsequence shift, packet subsequence duplication, and packet size variation. Based on the root causes of these three phenomenons, we propose TCP-aware traffic augmentation algorithms to generate potential flow variants.

Packet Subsequence Duplication Augmentation. The packet subsequence duplication is caused in the packet loss scenario where TCP uses RTO and fast retransmit mechanisms to handle packet loss. As their impact on the sequences of packet lengths is different, we design one traffic augmentation algorithm for each mechanism, respectively. The two algorithms incur packet subsequence duplication for each original TCP flow's packet length sequence by simulating the RTO and fast retransmit mechanisms with a packet loss rate, respectively. By setting different packet loss rates, the two algorithms can output massive TCP flow's packet length sequences containing packet subsequence duplication for each original TCP flow's packet length sequence. Due to space limitation, we put the algorithm details in the technical report [1].

Packet Subsequence Shift Augmentation. Similarly, the RTO and fast retransmit mechanisms can also lead to packet subsequence shift. Hence, we design two packet subsequence shift augmentation algorithms with the RTO and fast retransmit mechanisms, respectively. The two algorithms incur packet subsequence shift for each original TCP flow's packet length sequence by simulating the RTO and fast retransmit mechanisms with a packet loss rate, respectively. By setting different packet loss rates, the two algorithms can output massive TCP flow's packet length sequences containing packet subsequence shift for each original TCP flow's packet length sequence. Please refer to our additional technical report [1] for details.

Packet Size Variation Augmentation. The delays between TCP endpoints and the MTU will lead to packet size variation of TLS flows in different network environments. For a given RTT and an MSS, all the data segments sent during the RTT will be buffered in the TCP stack until the sender receives an ACK packet from the receiver or the size of the TCP stack is more than the MSS. Consequently, several data segments will be merged to generate a large packet rather than generate many small packets. Hence, we design a packet size variation augmentation algorithm by simulating a possible TCP transmission with an MSS and an RTT distribution. By setting different MSS values and RTT distributions, the algorithm can generate massive TCP flow's packet length sequences containing packet size variation for each original TCP flow's packet length sequence. Please refer to our additional technical report [1] for details.

4.4 Traffic Invariant Extractor

Rosetta uses a classical self-supervised learning approach, BYOL [24], to extract similar and robust features from flow variants that come from a flow. BYOL reduces the distance between positive pairs and applies the skillfully-designed momentum mechanism to prevent the hidden space from collapsing without negative pairs [9, 24]. BYOL relies on the *online* and *target* neural networks. The online network is defined by a set of parameters α and consists of three components: an *encoder* φ_α, a *projector* ψ_α, and a *predictor* p_α. For more details of the network structures, please refer to the study [24]. The target network is the same as the online network, except with a different set of parameters ζ. Given a target decay rate $\gamma \in [0,1]$, BYOL performs $\zeta \leftarrow \gamma\zeta + (1-\gamma)\alpha$ after each training update. It learns an encoder φ_α that outputs representations of input samples as k_α.

We use BYOL to project the flow variants into a hidden space as feature vectors and update the hidden space to minimize the distance among feature vectors of flow variants coming from the same flow. Specifically, two different flow variants are needed as the inputs of the online and target networks during the training phase. As TCP-aware traffic augmentation algorithms have configurable parameters, we can obtain a set of flow variants V from the packet length sequence of each flow s with different parameters. Here, we use v and v' to denote two flow variants in V. The online network outputs a representation $k_\alpha = \varphi_\alpha(v)$ and a projection $m_\alpha = \psi_\alpha(k)$ for v. Meanwhile, the target network outputs a representation $k'_\zeta = \varphi_\zeta(v')$ and a target projection $m'_\zeta = \psi_\zeta(k')$ from v'. Finally, the predictor p_α outputs a prediction $p_\alpha(m_\alpha)$ of m'_ζ. BYOL normalizes $p_\alpha(m_\alpha)$ and m'_ζ, and applies their mean squared error as the following loss function:

$$\mathcal{L}_{\alpha,\zeta} = \| \overline{p_\alpha}(m_\alpha) - \overline{m'_\zeta} \|_2^2 = 2 - 2 \cdot \frac{< p_\alpha(m_\alpha), m'_\zeta >}{\| p_\alpha(m_\alpha) \|_2 \cdot \| m'_\zeta \|_2} \quad (1)$$

The loss $\mathcal{L}_{\alpha,\zeta}$ is symmetrized in Eq. 1 by separately inserting v' into the online network and v into the target network to calculate $\widetilde{\mathcal{L}}_{\alpha,\zeta}$. In each training step, $\mathcal{L}^{Rosetta}_{\alpha,\zeta} = \mathcal{L}_{\alpha,\zeta} + \widetilde{\mathcal{L}}_{\alpha,\zeta}$ is minimized by optimizing α. After the training, we can use the encoder of BYOL φ_α to extract traffic invariants from packet length sequences of flows.

5 System Evaluation

We evaluate the effectiveness of Rosetta on improving the performance of the deep learning models for TLS encrypted

Table 6: Results on Classifying Replayed TLS Flows from CIRA-CIC-DoHBrw-2020 in Different Network Environments.

Model	Different Wired Network Environments					
	θ_1		θ_2		θ_3	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	93.05% (↓5.16%)	93.03%(↓5.17%)	82.00% (↑28.84%)	81.78%(↑46.87%)	83.72% (↑26.68%)	82.85%(↑46.53%)
SDAE + Rosetta	91.89% (↑0.42%)	91.77%(↑0.30%)	86.63% (↑30.42%)	86.63%(↑43.23%)	84.17% (↑28.42%)	83.69%(↑47.65%)
LSTM + Rosetta	86.63% (↓1.05%)	84.03%(↓3.44%)	79.89% (↑26.84%)	78.32%(↑43.25%)	82.00% (↑24.96%)	78.98%(↑42.41%)
DF + Rosetta	94.42% (↓4.00%)	94.39%(↓4.02%)	86.63% (↑33.37%)	86.63%(↑51.88%)	86.01% (↑27.98%)	85.83%(↑49.11%)
FS-Net + Rosetta	89.26% (↓1.48%)	89.12%(↓1.59%)	84.63% (↑23.47%)	84.47%(↑32.37%)	84.17% (↑26.07%)	83.50%(↑43.84%)
Transformer + Rosetta	94.11%(↓4.17%)	93.74%(↓2.26%)	84.11% (↑21.89%)	83.60%(↑29.48%)	83.37% (↑26.33%)	80.38%(↑38.38%)
On Average	**91.56%(↓2.57%)**	**91.01%(↓2.70%)**	**83.98%(↑27.47%)**	**83.57%(↑41.18%)**	**83.91%(↑26.74%)**	**82.54%(↑44.65%)**
Model	**Different Wireless Access Network Environments**					
	θ_4		θ_5		θ_6	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	89.05% (↑1.58%)	88.93%(↑1.90%)	85.37% (↑10.95%)	85.08%(↑13.55%)	80.42% (↑27.16%)	80.37%(↑45.41%)
SDAE + Rosetta	89.89% (↑1.78%)	89.74%(↑1.71%)	83.47% (↑1.36%)	82.95%(↑1.52%)	81.89% (↑26.73%)	81.88%(↑40.15%)
LSTM + Rosetta	85.37% (↑3.37%)	82.34%(↑1.15%)	82.53% (↑11.69%)	78.22%(↑10.87%)	76.53% (↑22.95%)	73.42%(↑37.33%)
DF + Rosetta	86.84% (↓1.16%)	86.53%(↓1.05%)	82.11% (↑7.16%)	81.31%(↑9.14%	82.63% (↑29.26%)	82.57%(↑47.57%)
FS-Net + Rosetta	85.58% (↓3.26%)	85.16%(↓3.60%)	84.42% (↑0.74%)	83.89%(↑0.60%)	77.26% (↑20.42%)	76.97%(↑32.47%)
Transformer + Rosetta	90.74% (↓3.00%)	89.81%(↓1.54%)	89.16% (↑3.54%)	88.20%(↑5.08%)	81.47% (↑27.20%)	79.63%(↑32.06%)
On Average	**87.91%(↓0.12%)**	**87.09%(↓0.24%)**	**84.51%(↑5.91%)**	**83.27%(↑6.79%)**	**80.03%(↑25.62%)**	**79.14%(↑39.17%)**

traffic classification in diverse network environments.

5.1 Experimental Setup

We use the same experiment setup mentioned in Section 3.1.1. Besides, we leverage our TCP-aware traffic augmentation algorithms to generate a number of flow variants from a third-party TLS traffic dataset [2] from a well-known security company. We leverage the generated flow variants to train the traffic invariant extractor (TIE) of Rosetta. Here, the dataset consists of 502,109 malicious TLS flows and 613,419 benign TLS flows. Actually, any traffic dataset containing massive flows could be used to train our traffic invariant extractor with our TCP-aware traffic augmentation algorithms.

As the dataset is *disjoint* with the previous two datasets used for traffic classification in our experiments, i.e., CIRA-CIC-DoHBrw-2020 and ISCX-VPN, we do not feed any prior knowledge about the traffic classification dataset into Rosetta. Here, to generate different flow variants, we set variable ranges for each parameter in our algorithms: the packet loss rate p ranges from 0% to 10%, L_{min} ranges from 1 to 3, L_{max} ranges from 2 to 10, RTT_{min} ranges from 0 to 100 ms, RTT_{max} ranges from 10 to 200 ms, and MSS ranges from 500 to 1500 bytes. When augmenting a flow, we randomly apply one combination of our five TCP-aware traffic augmentation algorithms to simulate random network environment effects on the flow. By repeating the augmentation process on the same flow, we obtain about 120 augmented flows for each flow. We train our TIE with the GeForce RTX 2080. It is trained with SGD for 60 epochs. We set the batch size of the training steps as 1024. The weight decay rate is 0.0004 and the target decay rate is 0.996.

[2] For the research community to know its details, we have released the partial dataset in Github: `https://github.com/sunskyXX/Rosetta.git`. To download the full dataset, researchers can contact us via our email addresses and sign the agreement on preventing data misusing and privacy leakage.

5.2 Performance Improvement with Rosetta

5.2.1 Improvement with Replayed Traffic

Results with Replayed TLS encrypted flows. As Table 6 shows, the performance of all the models significantly improves in the three network environments: θ_2, θ_3, and θ_6. Particularly, there is 44.65% F1-score improvement on average in θ_3. Besides, we can see that the accuracy and F1-score of the models drop slightly in θ_1 and θ_4. This is reasonable since Rosetta only extracts robust features and omits unstable features. Nevertheless, we should note that the unstable features may make DL alone have good performance in the environments similar to the training environment. However, these features significantly degrade the performance of DL alone in different environments in most cases. However, DL+Rosetta can always achieve acceptable classification performance in diverse environments by extracting robust features. On average, more than 80% accuracy can be achieved in all the six different network environments. Rosetta improves about 30% accuracy at most on classifying TLS flows in diverse network environments. We also get similar results for TLS flows from ISCX-VPN. Please refer to Table 13 in Appendix C.

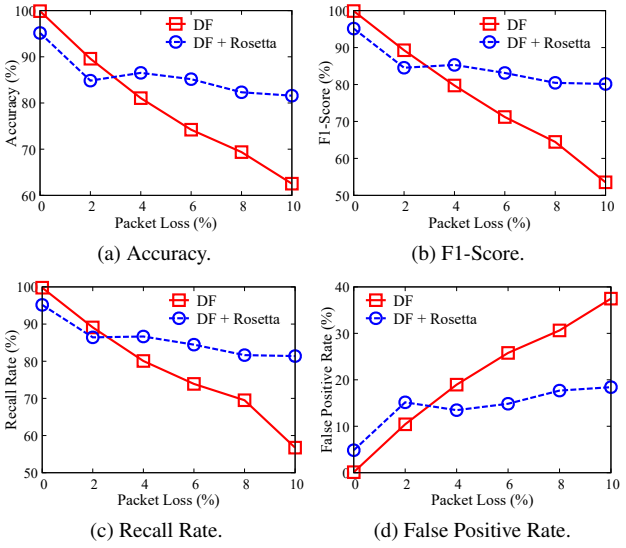
Table 7: Results on Classifying Real Website TLS Flows in Different Network Environments.

Model	Different Wired Network Environments					
	τ_1		τ_2		τ_3	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	86.63%(↓2.92%)	86.06%(↑4.19%)	84.83%(↑3.35%)	81.33%(↑0.45%)	91.04%(↑33.31%)	91.04%(↑38.75%)
SDAE + Rosetta	84.67%(↑2.30%)	81.54%(↑12.50%)	85.54%(↑7.41%)	84.49%(↑9.70%)	89.47%(↑19.43%)	85.87%(↑17.07%)
LSTM + Rosetta	84.17%(↑2.32%)	82.07%(↑5.48%)	76.01%(↓0.71%)	74.13%(↑0.05%)	88.52%(↑25.81%)	88.14%(↑30.89%)
DF + Rosetta	90.37%(↓0.90%)	90.10%(↑5.43%)	85.19%(↑1.24%)	81.00%(↑0.42%)	90.15%(↑6.56%)	90.14%(↑6.64%)
FS-Net + Rosetta	86.99%(↑1.18%)	86.47%(↑6.66%)	84.83%(↑11.81%)	76.24%(↑4.04%)	88.41%(↑23.99%)	88.40%(↑26.43%)
Transformer + Rosetta	90.02%(↑5.17%)	87.93%(↑1.74%)	85.36%(↑14.39%)	81.37%(↑11.80%)	89.70%(↑27.04%)	89.69%(↑31.23%)
On Average	**87.14%(↑1.19%)**	**85.69%(↑2.17%)**	**83.63%(↑6.25%)**	**79.76%(↑4.41%)**	**89.55%(↑22.69%)**	**88.88%(↑25.17%)**
Model	**Different Wireless Access Network Environments**					
	τ_4		τ_5		τ_6	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	77.24%(↑4.73%)	75.02%(↑6.51%)	83.58%(↑16.42%)	82.38%(↑22.23%)	75.92%(↑5.29%)	70.66%(↑1.93%)
SDAE + Rosetta	79.10%(↑11.06%)	77.54%(↑9.56%)	74.31%(↑9.74%)	71.18%(↑6.98%)	71.95%(↑2.01%)	65.80%(↑1.79%)
LSTM + Rosetta	69.28%(↑8.39%)	79.53%(↑19.49%)	75.16%(↑8.23%)	74.37%(↑10.77%)	69.59%(↑3.18%)	62.84%(↑1.17%)
DF + Rosetta	84.13%(↑4.23%)	81.67%(↑6.67%)	84.19%(↑13.52%)	80.48%(↑13.56%)	77.58%(↑4.55%)	76.10%(↑5.93%)
FS-Net + Rosetta	77.95%(↑7.81%)	74.79%(↑6.40%)	75.95%(↑11.11%)	72.87%(↑7.45%)	72.83%(↑5.18%)	67.44%(↑0.96%)
Transformer + Rosetta	76.84%(↑13.13%)	74.80%(↑12.66%)	75.66%(↓3.32%)	72.60%(↓2.78%)	76.60%(↑15.23%)	70.23%(↑10.49%)
On Average	**77.42%(↑8.23%)**	**77.22%(↑10.21%)**	**78.14%(↑9.28%)**	**75.64%(↑9.70%)**	**74.08%(↑5.91%)**	**68.85%(↑3.71%)**

5.2.2 Improvement with Real Traffic

Results with Website TLS encrypted flows. As Table 7 shows, the average performance of the models effectively improves on classifying the website TLS flows in all the six network environments. Particularly, there are 25.17% F1-Score improvement on average in τ_3. Rosetta can improve 33.31% accuracy at most for a deep learning model on classifying website TLS flows in diverse network environments.

Results with Application TLS Flows. As Table 8 shows, the average performance of the models effectively improves on classifying application TLS flows in all the six network environments. Rosetta improve 45.07% F1-Score at most for LSTM on classifying TLS flows in τ_3. The experiment results demonstrate that Rosetta is an effective approach to enhance the robustness of DL models for TLS flow classification in diverse network environments.



(a) Accuracy. (b) F1-Score.

(c) Recall Rate. (d) False Positive Rate.

Figure 5: Robustness under different packet loss rates.

5.3 Classification Robustness

To quantitatively evaluate the classification robustness when applying Rosetta in different network environments, we leverage `Traffic Control (TC)` and `ifconfig` to set different packet loss rates, delays, and MTU in our sender hosts that replay flows. Hence, we can build controllable network environments. We replay TLS flows from CIRA-CIC-DoHBrw-2020 in these controllable network environments and apply the DF model with/without Rosetta to classify these flows.[3]

Under Different Packet Loss Rates. Figure 5 shows the DF performance with and without Rosetta for different packet loss rates. As Figure 5a shows, although DF alone achieves more than 99% accuracy with a low packet loss rate, the accuracy quickly decreases when the packet loss increases.

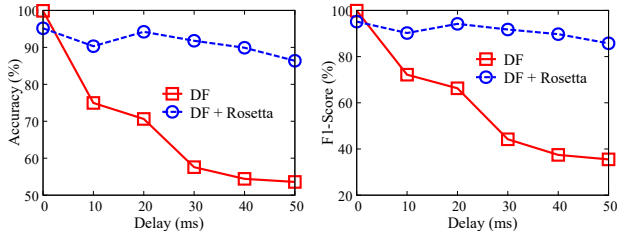
[3]We also conduct experiments with other classification models and they show similar results like DF. Hence, we only show the experimental results with DF for simplicity and clarity.

Table 8: Results on Classifying Real Application TLS Flows in Different Network Environments.

Model	Different Wired Network Environments					
	τ_1		τ_2		τ_3	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	90.62%(↓4.62%)	90.57%(↓4.50%)	80.97%(↑7.15%)	80.68%(↑9.29%)	81.24%(↓4.38%)	80.96%(↓4.17%)
SDAE + Rosetta	89.76%(↑8.09%)	89.40%(↑8.02%)	80.11%(↑26.75%)	76.28%(↑23.28%)	82.37%(↑23.62%)	82.30%(↑23.76%)
LSTM + Rosetta	92.58%(↑24.51%)	92.42%(↑35.29%)	79.74%(↑12.08%)	75.92%(↑34.34%)	83.58%(↑30.75%)	83.49%(↑45.07%)
DF + Rosetta	95.22%(↓2.89%)	95.06%(↓3.00%)	83.73%(↑0.26%)	79.76%(↑0.47%)	79.11%(↑9.16%)	79.04%(↑9.14%)
FS-Net + Rosetta	91.80%(↑22.18%)	91.55%(↑30.93%)	82.49%(↑11.70%)	78.53%(↑36.99%)	81.79%(↑25.43%)	81.64%(↑35.12%)
Transformer + Rosetta	96.98%(↑4.94%)	96.95%(↑5.60%)	84.12%(↓12.83%)	83.76%(↓12.70%)	84.56%(↓7.63%)	83.87%(↓8.24%)
On Average	**92.83%(↑8.70%)**	**92.66%(↑12.06%)**	**81.86%(↑7.52%)**	**79.15%(↑15.28%)**	**82.11%(↑12.83%)**	**81.88%(↑16.78%)**
Model	**Different Wireless Access Network Environments**					
	τ_4		τ_5		τ_6	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	97.76%(↑5.93%)	97.72%(↑6.19%)	97.44%(↑1.21%)	97.24%(↑1.33%)	96.18%(↑1.10%)	95.81%(↑1.24%)
SDAE + Rosetta	98.46%(↑21.94%)	98.44%(↑21.93%)	98.92%(↑26.21%)	98.86%(↑26.19%)	96.24%(↑37.05%)	95.91%(↑36.80%)
LSTM + Rosetta	97.59%(↑33.75%)	97.57%(↑42.22%)	98.78%(↑29.42%)	98.71%(↑41.12%)	94.76%(↑22.69%)	94.45%(↑33.41%)
DF + Rosetta	94.34%(↑1.38%)	94.21%(↑1.47%)	95.76%(↓2.93%)	95.43%(↓3.17%)	93.01%(↓5.98%)	92.35%(↓6.56%)
FS-Net + Rosetta	98.11%(↑38.48%)	98.09%(↑52.38%)	99.10%(↑30.85%)	99.05%(↑44.90%)	96.31%(↑25.25%)	96.05%(↑37.73%)
Transformer + Rosetta	93.28%(↑3.06%)	93.22%(↑3.19%)	99.75%(↑6.31%)	99.74%(↑7.28%)	93.45%(↑1.02%)	92.75%(↑0.42%)
On Average	**96.59%(↑17.42%)**	**96.54%(↑21.23%)**	**98.29%(↑15.18%)**	**98.17%(↑19.61%)**	**94.99%(↑13.52%)**	**94.55%(↑17.17%)**

When the packet loss is increased to 10%, DF only achieves 78% accuracy. High packet loss rates significantly change the pattern of traffic traces and make the traffic hard to be classified. However, the classification accuracy only slightly drops with high packet loss rates when enabling the DF model with Rosetta. The reason is that DF without Rosetta overfits the training network environment. It extracts some unstable features for better performance in environments similar to the training environment, e.g., the good results in the first two ticks in Figure 5a. However, these features significantly degrade the performance of DF alone in other environments. The experiments in a low delay environment and a high MTU environment show similar results. Figure 5b and Figure 5c show similar results for the F1-score and recall rate. As Figure 5d shows, although DF alone achieves less than 1% false positive rate with a low packet loss, the false positive rate quickly increases when packet loss increases. When the packet loss is increased to 10%, DF achieves 37% false positive rate. However, the false positive rate only slightly increase with high packet loss when enabling Rosetta. These results show that Rosetta enables deep learning models to achieve more robust traffic classification for different packet loss rates.

Under Different Delays. Figure 6 shows the DF performance with and without Rosetta for different delays. As Figure 6a shows, the DF model can achieve more than 95% accuracy when the delay is less than 5 ms. However, the accuracy significantly drops when the delay increases. Particularly, the accuracy drops to about 55% when the delay increases to 50 ms. However, when enabling the DF model with Rosetta, it can always achieve more than 86% accuracy with the delay

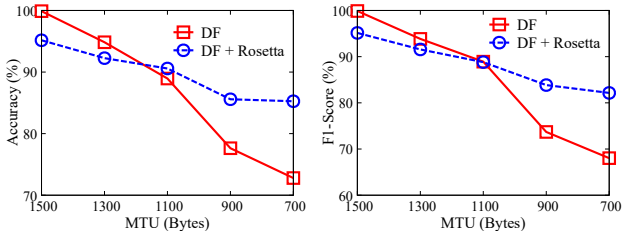


(a) Accuracy. (b) F1-Score.

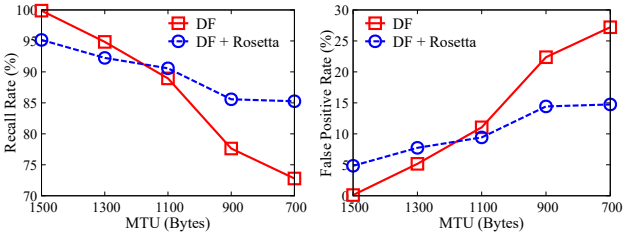
(c) Recall Rate. (d) False Positive Rate.

Figure 6: Classification robustness under different delays.

ranging from 10ms to 50ms. Besides, Figure 6b shows the DF model with Rosetta can always reach more than 84% F1-score for different delays. However, the F1-score remarkably drops with the increase of delays if Rosetta is not enabled. Figure 6c shows the similar results for recall rate. Figure 6d shows that DF with Rosetta keeps a low false positive rate for different delays. However, if Rosetta is disabled, the false positive rate significantly increases with the delays. The results show that Rosetta enables deep learning models to achieve more robust traffic classification for different delays.

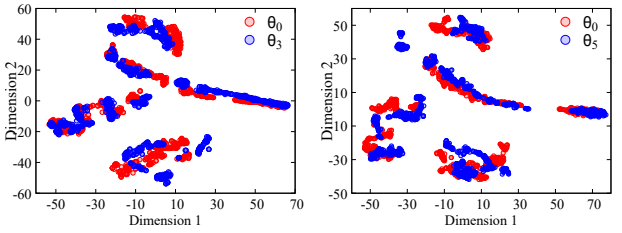


(a) Accuracy. (b) F1-Score.



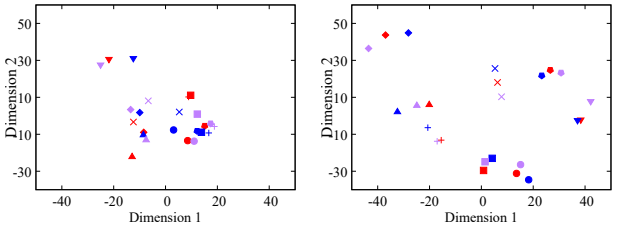
(c) Recall Rate. (d) False Positive Rate.

Figure 7: Classification robustness under different MTUs.



(a) Distribution between θ_0 and θ_3. (b) Distribution between θ_0 and θ_5.

Figure 8: Feature Distribution between two environments.



(a) DF (b) Rosetta

Figure 9: Feature vectors of eight flows. Each flow is replayed in three network environments, generating three points with the same shape but different colors.

Under Different MTUs. Figure 7 shows the DF performance with and without Rosetta for different MTUs. The performance of the DF model significantly decreases with the decrease of the MTU. Particularly, the F1-Score decreases from 99% to less than 70% when the MTU decreases from 1500 to 700 bytes. This is because a small MTU can cause packet size variation as we analyzed in Section 3.3, which causes significant pattern changes of packet sequences of flows. However, if we enable the DF model with Rosetta, the accuracy and F1-score become more stable for different MTUs. Even though the MTU drops to 700 bytes, we can still achieve more than 85% accuracy and 80% F1-score. The results show that Rosetta enables deep learning models to achieve more robust traffic classification for different MTUs.

5.4 Feature Vector Visualization

Rosetta generates a feature vector for each flow to denote the extracted abstract features in hidden features. Hence, we collect a pair of feature vectors for each flow that is replayed in two different network environments. We apply the t-SNE method to project these high-dimensional feature vectors into 2D vectors. Figure 8 draws all the feature vectors in two different environments. We can see that the distributions of feature vectors are similar in different network environments. These results demonstrate that Rosetta extracts stable and robust features for flows. Figure 9 further demonstrates the results. As shown in Figure 9b, Rosetta generates similar feature vectors for a flow in different network environments. Besides, feature vectors of different flows are separated. Nevertheless, the DF model generates nearly overlapped feature vectors for different flows, which is shown in Figure 9a. Feature vectors for a flow in different network environments are separated.

5.5 Other Data Augmentation Methods

We conduct experiments to compare our TCP-aware traffic augmentation method with two classical data augmentations: Random Mask (RM) [17] and Random Swap (RS) [60]. We replace our augmentation method with RM and RS in Rosetta to augment flows, respectively. Hence, we get three versions of Rosetta with different traffic augmentation methods. Based on the feature vectors extracted by the three versions of Rosetta, we train three DF models to classify TLS flows from CIRA-CIC-DoHBrw-2020 in different network environments, respectively. Table 9 shows the classification results with different data augmentations in diverse network environments. We can only get 47.73% F1-score on average in θ_0-θ_6 if RM is applied to augment flows in Rosetta. Similarly, the average F1-score with RS in θ_0-θ_6 is still less than 50%. However, if we apply our TCP-aware traffic augmentation in Rosetta, we can achieve 87.49% F1-Score in θ_0-θ_6 on average. These results demonstrate that simply applying data augmentation methods without considering TCP semantics cannot help Rosetta extract robust features in different network environments.

6 Discussion

Considerations on Reconstructing Packets before Training Classifiers. We may reconstruct out-of-order segments packets and discard the duplicated packets by keeping track of the sequence numbers for each flow; however, it is difficult to reconstruct reassembly packets due to the Nagle algorithm

Table 9: Classification Results with Different Data Augmentations in Different Network Environments.

Data Aug.	Different Wired Network Environments								Different Wireless Access Network Environments						On Average	
	θ_0		θ_1		θ_2		θ_3		θ_4		θ_5		θ_6			
	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
RM [17]	97.89%	97.80%	89.47%	88.12%	53.26%	11.56%	58.03%	16.47%	78.00%	71.72%	61.58%	34.00%	52.84%	14.44%	70.15%	47.73%
RS [60]	99.79%	99.77%	86.42%	83.09%	56.26%	16.16%	56.13%	21.84%	77.47%	68.53%	58.53%	20.88%	53.16%	16.74%	69.68%	46.72%
Ours	95.16%	95.14%	94.42%	94.39%	86.63%	86.63%	86.01%	85.83%	86.84%	86.53%	82.11%	81.31%	82.63%	82.57%	**87.69%**	**87.49%**

or the fragmented packets due to the various MTUs in different network environments. The Nagle algorithm (enabled by default in major TCP implementations [61]) merges packet payloads based on the network environments and application behaviors, which cannot be obtained from the packets. Similarly, how the original data are fragmented into different packets is based on MTUs and application behaviors, which cannot be obtained from the packets and flows. Besides, reconstructing out-of-order packets or dropping duplicated packets requires keeping track of the TCP state for each flow. It is resource-consuming and time-consuming when there are massive TLS flows in real networks, especially for the real-time traffic classification tasks [5, 6]. Instead, Rosetta extracts robust flow features, enabling accurate traffic classification in diverse network environments without reconstructing packets.

Considerations on Periodically Re-training Classifiers. We may re-train a classifier in the target network periodically for performance improvement with diverse network environments. However, there may still be significant performance degradation for the classifier in the network environment that the classifier hasn't met before, especially considering there are time-varying and infinite environmental states for a real network. It is also almost impossible to train a classifier that covers all the possible network environments of the target network. Besides, periodically re-training the classifier requires frequently collecting new flows in new network environments and labeling them with humans. The procedure is resource-consuming, time-consuming, and labor-intensive in practice. It is also hard to determine when to retrain the classifier. Thus, it is necessary to achieve robust TLS traffic classification in diverse network environments without re-training the models.

Robustness-Accuracy Tradeoff with Rosetta. Even though our experiments show Rosetta can enable DL models to achieve robust TLS encrypted traffic classification in diverse network environments, we observe that Rosetta may cause slight performance degradation for DL models in a few cases. This is likely to happen when the network environment where we classify TLS flows is similar to the network environments where we train our DL models. It is reasonable since Rosetta extracts robust features and omits unstable features. However, Rosetta can always enable DL models to achieve acceptable classification performance in diverse environments with the robust features. Instead, DL models without Rosetta may overfit the training network environment. They extract some unstable features for better performance in the environments similar to the training environment. However, these features significantly degrade the performance of DL alone in most environments as we have shown in Section 3. Rosetta enables robust TLS flow classification in diverse network environments at the cost of slight accuracy drop in a few cases. Considering the varying real-world network environments, we believe that Rosetta makes an acceptable tradeoff between classification robustness and accuracy.

Considerations on Timing Information. Rosetta does not extract features from timing information of TLS flows. Timing information may be useful for website fingerprinting [44]. However, we focus on general TLS traffic classification in a real network with diverse network environments. Different network environments can significantly change the timing information of flows. For example, timing in a high-delay network environment is different from that in a low-delay network environment. Consequently, the timing information can incur potential performance degradation for traffic classification in diverse network environments. Hence, Rosetta does not extract features from timing information to classify TLS flows. Actually, recent studies [46, 51, 52] have dropped the timing information to achieve excellent performance for traffic classification.

7 Conclusion

In this paper, we conduct a comprehensive measurement study to demonstrate that existing DL models fail to achieve robust TLS traffic classification in diverse network environments. Through our systematic analysis, we point out the reason for the remarkable performance degradation is that the packet length sequences of flows dramatically change due to different mechanisms of TCP for reliable transmission in diverse network environments. To address the problem, we develop Rosetta to learn robust features of TLS flows in diverse network environments based on TCP-aware traffic augmentations and self-supervised learning. Extensive experiments demonstrate that Rosetta can enable robust TLS encrypted traffic classification in diverse network environments for existing DL models and significantly improve their classification performance. We hope this work attracts more attention on TLS traffic classification considering real network environments.

Acknowledgments

We are grateful to all the anonymous reviewers for their insightful comments. The research is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62132011, 62002192, 62202260, 62221003, and 61832013, the NSA under Grant H98230-22-1-0311, the China Postdoctoral Science Foundation under Grant 2022M721824 and 2022M720202, the Shuimu Tsinghua Scholar Program, and the Beijing Postdoctoral Research Foundation under Grant 2022-ZZ-078. Jiahao Cao and Mingwei Xu are the corresponding authors of the paper. Thanks for Yixiao Wang's contribution to the experiments.

References

[1] Additional technical and experimental report for rosetta. https://cloud.tsinghua.edu.cn/f/7f250d2ffce8404b845e/?dl=1.

[2] Mohammed Alfatafta, Basil Alkhatib, Ahmed Alquraan, and Samer Al-Kiswany. Toward a generic fault tolerance technique for partial network partitioning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 351–368, 2020.

[3] Blake Anderson and David McGrew. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, pages 1723–1732, 2017.

[4] Ons Aouedi, Kandaraj Piamrat, and Dhruvjyoti Bagadthey. A semi-supervised stacked autoencoder approach for network traffic classification. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2020.

[5] Jiasong Bai, Menghao Zhang, Guanyu Li, Chang Liu, Mingwei Xu, and Hongxin Hu. Fastfe: Accelerating ml-based traffic analysis with programmable switches. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, pages 1–7, 2020.

[6] Diogo Barradas, Nuno Santos, Luıs Rodrigues, Salvatore Signorello, Fernando MV Ramos, and André Madeira. Flowlens: Enabling efficient flow classification for ml-based network security applications. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, 2021.

[7] Jiahao Cao, Zijie Yang, Kun Sun, Qi Li, Mingwei Xu, and Peiyi Han. Fingerprinting {SDN} applications via encrypted control traffic. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*, pages 501–515, 2019.

[8] Lionel F Gonzalez Casanova and Po-Chiang Lin. Generalized classification of dns over https traffic with deep learning. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1903–1907. IEEE, 2021.

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[10] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 753–770, 2022.

[11] Niels Christensen, Mark Glavind, Stefan Schmid, and Jiří Srba. Latte: improving the latency of transiently consistent network update schedules. *ACM SIGMETRICS Performance Evaluation Review*, 48(3):14–26, 2021.

[12] Cisco. Cisco encrypted traffic analytics. Technical report, 2021.

[13] Mauro Conti, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security*, 11(1):114–125, 2015.

[14] Scott E Coull and Kevin P Dyer. Traffic analysis of encrypted messaging services: Apple imessage and beyond. *ACM SIGCOMM Computer Communication Review*, 44(5):5–11, 2014.

[15] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[16] Tianyu Cui, Gaopeng Gou, Gang Xiong, Zhen Li, Mingxin Cui, and Chang Liu. {SiamHAN}:{IPv6} address correlation attacks on {TLS} encrypted traffic via siamese heterogeneous graph attention network. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4329–4346, 2021.

[17] 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, NAACL-HLT*, 2019.

[18] Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[19] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414. sn, 2016.

[20] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. Realtime robust malicious traffic detection via frequency domain analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3431–3446, 2021.

[21] Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, 2019.

[22] Nan Geng, Tian Lan, Vaneet Aggarwal, Yuan Yang, and Mingwei Xu. A multi-agent reinforcement learning perspective on distributed traffic engineering. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2020.

[23] Roberto Gonzalez, Claudio Soriente, and Nikolaos Laoutaris. User profiling in the time of https. In *Proceedings of the 2016 Internet Measurement Conference*, pages 373–379, 2016.

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[25] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] Jordan Holland, Paul Schmitt, Nick Feamster, and Prateek Mittal. New directions in automated traffic analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3366–3383, 2021.

[28] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[29] Guoliang Kang, Xuanyi Dong, Liang Zheng, and Yi Yang. Patchshuffle regularization. *arXiv preprint arXiv:1707.07103*, 2017.

[30] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2017.

[31] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, 2018.

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[33] XuKui Li, Wei Chen, Qianru Zhang, and Lifa Wu. Building auto-encoder intrusion detection system based on random forest feature selection. *Computers & Security*, 95:101851, 2020.

[34] 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*, pages 633–642, 2022.

[35] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li. Fs-net: A flow sequence network for encrypted traffic classification. In *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*, pages 1171–1179. IEEE, 2019.

[36] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, 2020.

[37] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[38] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[39] Mohammadreza MontazeriShatoori, Logan Davidson, Gurdip Kaur, and Arash Habibi Lashkari. Detection of doh tunnels using time-series classification of encrypted traffic. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 63–70. IEEE, 2020.

[40] Francisco J Moreno-Barea, Fiammetta Strazzera, José M Jerez, Daniel Urda, and Leonardo Franco. Forward noise adjustment scheme for data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)*, pages 728–734. IEEE, 2018.

[41] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *NDSS*, 2016.

[42] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[43] Selenium Project. Selenium webdriver, 2022.

[44] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. Tik-tok: The utility of packet timing in website fingerprinting attacks. *arXiv preprint arXiv:1902.06421*, 2019.

[45] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. *ACM SIGCOMM Computer Communication Review*, 42(4):323–334, 2012.

[46] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[47] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[48] Danfeng Shan, Fengyuan Ren, Peng Cheng, Ran Shu, and Chuanxiong Guo. Observing and mitigating microburst traffic in data center networks. *IEEE/ACM Transactions on Networking*, 28(1):98–111, 2019.

[49] Meng Shen, Yiting Liu, Liehuang Zhu, Xiaojiang Du, and Jiankun Hu. Fine-grained webpage fingerprinting using only packet length information of encrypted traffic. *IEEE Transactions on Information Forensics and Security*, 16:2046–2059, 2020.

[50] Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, and Xiaojiang Du. Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. *IEEE Transactions on Information Forensics and Security*, 16:2367–2380, 2021.

[51] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1928–1943, 2018.

[52] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1131–1148, 2019.

[53] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):63–78, 2017.

[54] Ying Tian, Zhiliang Wang, Xia Yin, Xingang Shi, Yingya Guo, Haijun Geng, and Jiahai Yang. Traffic engineering in partially deployed segment routing over ipv6 network with deep reinforcement learning. *IEEE/ACM Transactions on Networking*, 28(4):1573–1586, 2020.

[55] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[56] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J Dubois, Martina Lindorfer, David Choffnes, Maarten van Steen, and Andreas Peter. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Network and Distributed System Security Symposium (NDSS)*, volume 27, 2020.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[58] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon

Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[59] Xin Wang, Shuhui Chen, and Jinshu Su. App-net: a hybrid neural network for encrypted mobile traffic classification. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 424–429. IEEE, 2020.

[60] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing EMNLP-IJCNLP*, 2019.

[61] Wikipedia. Nagle's algorithm. https://en.wikipedia.org/wiki/Nagle's_algorithm, Octorber 2022.

[62] Junchi Xing and Chunming Wu. Detecting anomalies in encrypted traffic via deep dictionary learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 734–739. IEEE, 2020.

[63] Ao Yu, Hui Yang, Kim Khoa Nguyen, Jie Zhang, and Mohamed Cheriet. Burst traffic scheduling for hybrid e/o switching dcn: An error feedback spiking neural network approach. *IEEE Transactions on Network and Service Management*, 18(1):882–893, 2020.

[64] Yinbo Yu, Xing Li, Xue Leng, Libin Song, Kai Bu, Yan Chen, Jianfeng Yang, Liang Zhang, Kang Cheng, and Xin Xiao. Fault management in software-defined networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(1):349–392, 2018.

[65] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. Homonit: Monitoring smart home apps from encrypted traffic. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1074–1088, 2018.

[66] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

A Details for Transformer

We use a standard encoder and MLP in Transformer as a classifier in our experiments. As the input of the encoder in Transformer is a token sequence, we treat the packet length sequence as the token sequence for the input. Following the general settings of existing methods on traffic classification [51, 52], we fix the length of the input sequence to 100 for the Transformer encoder, i.e., any sequence longer than 100 will be truncated to 100, and any sequence shorter than 100 is padded with zeros. Thus, the encoder output will generate embedding vectors with a fixed size. Next, we flatten the sequence of embedding vectors into a new embedding vector as the input for MLP. We train Transformer with massive packet length sequences from TLS flows and we use SGD to optimize its parameters. We optimize its hyperparameters according to the classification results and our experience. The hyperparameters of Transfomer are shown in Table 10.

Table 10: Hyperparameters of our Transformer.

Hyperparameter	Trasnformer
hidden dimension	16
layer	4
heads	4
dropout rate	0.1
optimizer	SGD
batch size	32
training epoch	30
learning rate	0.0003

Table 11: Ablation Study on Different Modules in Rosetta.

Alg.1	Alg.2	Alg.3	Alg.4	Alg.5	TIE	AC	F1
×	×	×	×	×	×	58.03%	36.72%
×	✓	✓	✓	✓	✓	83.03%	83.28%
✓	×	✓	✓	✓	✓	82.57%	82.50%
✓	✓	×	✓	✓	✓	82.80%	82.21%
✓	✓	✓	×	✓	✓	84.29%	84.09%
✓	✓	✓	✓	×	✓	60.42%	41.32%
✓	✓	✓	✓	✓	×	72.57%	78.41%
✓	✓	✓	✓	✓	✓	**86.01%**	**85.83%**

B Ablation Study

We conduct ablation experiments for the five traffic augmentation algorithms and the Traffic Invariant Extractor (TIE) in Rosetta. We train different versions of Rosetta by removing one of the six modules each time. We train different DF models with different versions of Rosetta to classify TLS flows from CIRA-CIC-DoHBrw-2020 in diverse network environments. Note that, the version of Rosetta without TIE indicates that it only conducts TCP-aware traffic augmentation, and we use augmented traffic to directly train classifiers without using TIE to extract features. Table 11 shows the classification results for our ablation experiments. The classification accu-

Table 12: Results on Classifying Replayed TLS Flows from ISCX-VPN in Different Network Environments (Without Rosetta).

Model	Different Wired Network Environments								Different Wireless Access Network Environments					
	θ_0		θ_1		θ_2		θ_3		θ_4		θ_5		θ_6	
	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
CNN	76.92%	76.83%	71.53%	72.65%	60.90%	60.43%	59.62%	56.62%	53.85%	52.53%	57.33%	56.96%	67.95%	67.86%
SDAE	77.56%	77.56%	66.23%	65.39%	59.62%	59.60%	56.41%	56.38%	52.56%	49.57%	54.67%	53.48%	61.54%	61.48%
LSTM	82.69%	82.43%	66.23%	65.53%	62.18%	57.24%	61.54%	55.85%	62.82%	60.97%	62.67%	59.15%	64.74%	60.14%
DF	81.41%	81.37%	72.73%	72.35%	64.10%	63.89%	64.74%	64.73%	61.54%	61.31%	59.33%	59.03%	58.97%	57.61%
FS-Net	78.85%	78.53%	68.18%	68.15%	51.92%	51.59%	47.44%	46.37%	51.28%	51.08%	60.00%	59.89%	54.49%	53.94%
Transformer	82.14%	84.85%	72.08%	71.54%	62.95%	61.77%	56.41%	58.46%	58.97%	61.31%	64.67%	61.55%	62.82%	63.15%
On Average	**79.93%**	**80.26%**	**69.50%**	**69.29%**	**60.28%**	**59.09%**	**57.69%**	**56.40%**	**56.84%**	**56.13%**	**59.78%**	**58.34%**	**61.75%**	**60.70%**

Table 13: Results on Classifying TLS Flows from ISCX-VPN in Different Network Environments (With Rosetta).

Model	Different Wired Network Environments					
	θ_1		θ_2		θ_3	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	77.27%(↑5.74%)	76.84%(↑4.19%)	74.84%(↑13.94%)	74.43%(↑14.00%)	71.79%(↑12.17%)	70.21%(↑13.59%)
SDAE + Rosetta	78.57%(↑12.34%)	77.89%(↑12.50%)	76.28%(↑16.66%)	75.17%(↑15.57%)	75.64%(↑19.23%)	74.57%(↑18.19%)
LSTM + Rosetta	74.03%(↑7.80%)	71.01%(↑5.48%)	76.28%(↑14.10%)	70.87%(↑13.62%)	75.00%(↑13.46%)	71.11%(↑15.26%)
DF + Rosetta	78.57%(↑5.84%)	77.78%(↑5.43%)	75.64%(↑11.54%)	74.83%(↑10.94%)	77.56%(↑12.82%)	77.07%(↑12.34%)
FS-Net + Rosetta	75.32%(↑7.14%)	74.81%(↑6.66%)	75.00%(↑23.08%)	74.54%(↑22.95%)	76.92%(↑29.48%)	76.45%(↑30.09%)
Transformer + Rosetta	76.27%(↑4.19%)	73.28%(↑1.74%)	76.92%(↑13.97%)	73.13%(↑11.36%)	77.56%(↑21.15%)	75.18%(↑16.72%)
On Average	**76.67%(↑6.15%)**	**75.27%(↑5.14%)**	**75.83%(↑13.33%)**	**73.83%(↑12.64%)**	**75.75%(↑15.47%)**	**74.10%(↑15.17%)**

Model	Different Wireless Access Network Environments					
	θ_4		θ_5		θ_6	
	AC	F1	AC	F1	AC	F1
CNN + Rosetta	75.00%(↑21.15%)	73.97%(↑21.44%)	74.33%(↑17.00%)	72.06%(↑15.10%)	74.36%(↑6.41%)	73.23%(↑5.37%)
SDAE + Rosetta	76.92%(↑24.36%)	76.15%(↑26.58%)	75.33%(↑20.66%)	74.51%(↑21.03%)	78.21%(↑16.67%)	77.58%(↑16.10%)
LSTM + Rosetta	75.64%(↑12.82%)	71.21%(↑10.25%)	74.00%(↑11.33%)	69.77%(↑10.62%)	78.85%(↑14.11%)	75.91%(↑15.77%)
DF + Rosetta	76.28%(↑14.74%)	76.00%(↑14.69%)	73.33%(↑14.00%)	72.85%(↑13.83%)	78.21%(↑19.24%)	77.84%(↑20.23%)
FS-Net + Rosetta	71.15%(↑19.87%)	70.81%(↑19.73%)	70.67%(↑10.67%)	70.48%(↑10.59%)	71.79%(↑17.30%)	71.79%(↑17.85%)
Transformer + Rosetta	77.56%(↑18.59%)	74.45%(↑13.14%)	75.33%(↑10.66%)	71.76%(↑10.21%)	73.08%(↑10.26%)	68.18%(↑5.03%)
On Average	**75.43%(↑15.93%)**	**73.77%(↑15.12%)**	**73.83%(↑12.05%)**	**71.90%(↑11.62%)**	**75.75%(↑12.00%)**	**74.09%(↑11.48%)**

racy is ranging from 60.42% to 84.29% by removing one of the five traffic augmentation algorithms. We can see that any of the five algorithms is helpful to improve classification results in diverse network environments. However, the accuracy is only 60.42% without Algorithm 5. Hence, Algorithm 5 plays the most important role in improving the classification performance of TLS flows in diverse network environments. Besides, we can see that the accuracy is 72.57% without TIE, while the accuracy is 86.01% with TIE. Hence, applying TIE to extract robust features after TCP-aware traffic augmentation is necessary to further improve the classification performance.

C Results with TLS flows from ISCX-VPN

The experimental results with the replayed flows from ISCX-VPN in different network environments demonstrate similar results, which is shown in Table 12. When applying the models trained in θ_0 to conduct traffic classification in θ_1 to θ_6, the average accuracy of all the models decreases by about 10% at least and 23% at most, and the average F1-Score decreases by about 11% at least and 24% at most. Hence, it is obvious that different network environments can significantly affect the traffic classification results of existing deep models. The results demonstrate that existing deep learning models fail to enable robust TLS encrypted traffic classification in diverse network environments. However, Rosetta can effectively improve the classification robustness of the DL models in diverse network environments, which is shown in Table 13.