Toward Robust Detection of Malicious Encrypted Traffic Using Only Low-Quality Training Data

Yuqi Qing, *Student Member, IEEE*, Qilei Yin, Xinhao Deng, Yihao Chen, Zhuotao Liu, *Senior Member, IEEE*, Kun Sun, *Senior Member, IEEE*, Ke Xu, *Fellow, IEEE*, Jia Zhang, and Qi Li, *Senior Member, IEEE*

***Abstract*—Machine learning (ML) is promising in accurately detecting malicious flows in encrypted network traffic; however, it is challenging to collect a training dataset that contains a sufficient amount of encrypted malicious data with correct labels. When ML models are trained with low-quality training data, they suffer degraded performance. In this paper, we aim to address a real-world low-quality training dataset problem, namely, detecting encrypted malicious traffic generated by continuously evolving malware. We develop RAPIER+ that fully utilizes different distributions of normal and malicious traffic data in the feature space, where normal data is tightly distributed in a certain area, and the malicious data is scattered over the entire feature space to augment training data for model training. RAPIER+ includes two pre-processing modules to convert traffic into feature vectors and correct label noises. We evaluate our system on two public datasets and one combined dataset. With 1000 samples and 45% noise from each dataset, our system achieves the F1 scores of 0.78, 0.84, and 0.87, respectively, achieving average improvements of 358.5%, 314.0%, and 221.1% over the existing methods, respectively. Furthermore, we evaluate RAPIER+ with a real-world dataset obtained from a security enterprise. RAPIER+ effectively achieves encrypted malicious traffic detection with the best F1 score of 0.81 and improves the F1 score of existing methods by an average of 288.7%.**

***Index Terms*—Malicious encrypted traffic detection, machine learning, robust detection.**

Received 6 November 2025; revised 10 March 2026; accepted 24 April 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor C. Peng. Date of publication 5 May 2026; date of current version 26 May 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62132011, Grant 62425201, Grant 62472247, and Grant 62502473; in part by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China under Grant JYB2025XDXM114; and in part by Beijing–Tianjin–Hebei Natural Science Foundation Cooperation Project under Grant 25JJJC0003. A preliminary version of this manuscript has been published in the proceedings of the Network and Distributed System Security (NDSS) Symposium 2024 [DOI: 10.14722/ndss.2024.23081]. *(Corresponding author: Qi Li.)*

Yuqi Qing, Xinhao Deng, Zhuotao Liu, and Jia Zhang are with INSC, Tsinghua University, Beijing 100084, China (e-mail: qyq21@mails.tsinghua.edu.cn; xinhaodeng.thu@gmail.com; zhuotaoliu@tsinghua.edu.cn; zhangjia2017@tsinghua.edu.cn).

Qilei Yin is with the Zhongguancun Laboratory, Beijing 100094, China (e-mail: yinql@zgclab.edu.cn).

Yihao Chen is with DCST, Tsinghua University, Beijing 100084, China (e-mail: yh-chen21@mails.tsinghua.edu.cn).

Kun Sun is with IST, George Mason University, Fairfax, VA 22030 USA (e-mail: ksun3@gmu.edu).

Ke Xu is with DCST, Tsinghua University, Beijing 100084, China, also with the Zhongguancun Laboratory, Beijing 100094, China, and also with the State Key Laboratory of Internet Architecture, Tsinghua University, Beijing 100084, China (e-mail: xuke@tsinghua.edu.cn).

Qi Li is with INSC, Tsinghua University, Beijing 100084, China, also with the Zhongguancun Laboratory, Beijing 100094, China, and also with the State Key Laboratory of Internet Architecture, Tsinghua University, Beijing 100084, China (e-mail: qli01@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TON.2026.3690471

I. INTRODUCTION

MALICIOUS traffic detection [6], [22], [23], [24], [37], [61], [63], [69], [85], [106] have been extensively studied. As more malware start to use encryption protocols to hide traffic content, the traditional detection methods focusing on analyzing plaintext payloads are obsolete. Learning-based methods [2], [3], [26], [49], [50], [68], [71], [74], [80], [84], [92], [101], [105], [11], [15], [17], [52], [65], [94], [100] adopt machine-learning (ML) models to learn discriminative representations from encrypted traffic for detection. These designs are mostly supervised, heavily relying on a training dataset that contains high-quality data samples to build accurate and robust detection models.

It is non-trivial to collect high-quality training data. First, collecting time-sensitive malware data is difficult. The typical approach is to execute malware already captured in the real network in sandboxes and collect the traffic [58], [89]. However, since the malware evolves continuously, the captured malware have insufficient time sensitivity. For example, in the first quarter of 2023, approximately 70% of malware detected in unencrypted web traffic were zero-day threats [88], highlighting the prevalence of unknown malware in real network environments. Second, it isn't easy to correctly label all encrypted traffic in practice. Prior studies have shown that the network intrusion datasets collected with simple labeling methods, *e.g.* according to host labels or by public service, contain substantial label noise. For example, CIC-IDS-2017, which is labeled according to the attack-executing hosts, has been reported to contain approximately 25% noise in the benign set and nearly 50% noise in the malicious set [18]. In addition, a single anti-virus labeling service can introduce around 35.71% noise in the Windows PE Malware Dataset and 47.5% noise in the Android Malware Dataset [90]. Given the non-trivial overhead of manual labeling [31], [67], collected datasets inevitably contain label noise. Therefore, the training dataset collected in reality is typically limited in both quality and size.

Prior art [20], [62], [73], [76], [83], [86] shows that limited samples or label noises can lead to poor detection

2998-4157 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

performances. The issue becomes worse when the limited training samples also include label noises, *i.e.* low-quality training data. However, existing techniques like data augmentation [19], [37], [47], [98], [9] and robust ML models against label noises [10], [29], [32], [35], [45], [56], [62], [75], [78], [91], [93], [95], [96], [40], [41], [46], [51], [57], [97], [99] cannot address the issue. Specifically, the data augmentation methods synthesize new data based on existing data. However, label noises will confuse the distributions of different categories, resulting in new data being synthesized from an incorrect distribution, leading to more label noises. Further, to correct the impacts of label noises, existing art relies on strong assumptions and prior knowledge [10], [29], [35], [62], [78], [91], [95], [46] or a large-sized training set that can reveal each sample's intrinsic characteristics [32], [56], [75], [93], [96], [40], [41], [51], [57], [97], [99]. None of these prerequisites is satisfied in our problem. Recent encrypted traffic classification methods [14], [48], [104] improved the performance under limited labeled data by learning knowledge from large-scale unlabeled data. However, collecting such a large-scale dataset is expensive. It may also increase the risk of privacy leakage [39].

In this paper, we propose a robust encrypted malicious traffic detection system RAPIER+ to address the low-quality training data. The core idea is to leverage the difference between the distributions of normal and malicious traffic to estimate the possible region of each type. RAPIER+ does not depend on the correct labels or the amount of training data, thus it can be trained on a low-quality dataset that contains label noises. We observe that normal behaviors are typically more representative and consistent than malicious ones, making normal data tend to exhibit a denser distribution than malicious data. Thus, we can infer the true labels for training samples based on their data distributions. In addition, attackers may imitate normal behaviors by generating malicious data with a distribution similar to normal data over time. To address this case, we synthesize new training data located in the possible regions of new malicious data to improve the detection ability of the model.

RAPIER+ is powered by three tightly coupled components. First, we propose a Transformer-based feature extraction module to convert raw encrypted network traffic into feature vectors representing traffic's behaviors. Due to the Transformer model's ability to be trained in an unsupervised manner and its insensitivity to input data order, our feature extraction module can accurately capture the comprehensive behavior of encrypted traffic while mitigating the adverse effects of label noise in the low-quality training set and traffic distortion caused by dynamic network conditions. Second, we design a distribution-aware label noise correction module to infer the true labels of training data. It estimates the distribution of training data via an auto-regressive generative model, re-labels the data exhibiting the most obvious distribution characteristics, and infers the labels of the remaining data through a supervised Transformer. Third, we develop a new data augmentation module to synthesize new training data. This module selects the possible distribution regions of new malicious data and applies an improved Generative Adversarial Network (GAN) model to generate new training data in these target regions. The synthetic data is combined with the label-corrected data for training. In summary, our contributions are as follows:

- We develop a system called RAPIER+ to detect encrypted malicious traffic in the case that the amount of training data is limited and non-negligible label noises exist. It is a malware traffic detection system that simultaneously overcomes the challenges of both training data insufficiency and label noises.
- We implement RAPIER+ and perform extensive evaluations based on two public datasets and one combined dataset. With only 1000 training samples (*i.e.* 500 malicious and 500 normal ones) and 45% noise ratio, RAPIER+ effectively achieves the F1 scores of 0.78, 0.84, and 0.87 on the three datasets, respectively, achieving average improvements of 358.5%, 314.0%, and 221.1% over the existing methods.
- We evaluate RAPIER+ with a real-world dataset collected by a security enterprise. It achieves the best label correction performance when the noise ratios vary from 20% to 45%, consistently reducing the noise ratios to less than 4.0%. Furthermore, it can effectively achieve encrypted malicious traffic detection with the best F1 score of 0.81 and improve the F1 score of existing methods by an average of 288.7%.

II. Background

A. Data Augmentation

Data augmentation techniques can effectively increase the size of the training data set without explicitly collecting new data. The most common data-augmentation strategy is over-sampling. In particular, the new data can be randomly sampled or synthesized based on random combinations of existing data [8], [33], [34], [38]. As a result, the ML model may overfit the training set due to the limited diversity of the data. Recently, the Generative Adversarial Network (GAN) [30] has been widely applied for data augmentation in various areas, including malicious encrypted traffic detection [9]. Powered by an adversarial framework, GAN may learn the distribution of existing data and then create a variety of new training data that conform to the distribution. However, these methods are not resilient to label noises. The false labels will confuse the data distribution of different categories, leading to more label noises in new samples. Thus, the existing data augmentation methods cannot handle the low-quality issue of encrypted training data.

B. Robust Machine Learning Models

Robust machine learning models aim to mitigate the impacts of incorrectly labeled training data. Several models [28], [35], [55], [62], [87], [91], [102] apply the label transition matrix, recording the probability of a category being mislabeled into another category. However, the label transition matrix is usually unknown in prior. Recent studies [4], [32], [56], [75], [40], [41], [46], [51], [57], [97], [99] address the issue by automatically identifying mislabeled samples based on their properties, *e.g.*, higher loss value in the training stage. However, these properties may not hold when the number of training samples decreases, but may significantly constrain these methods.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

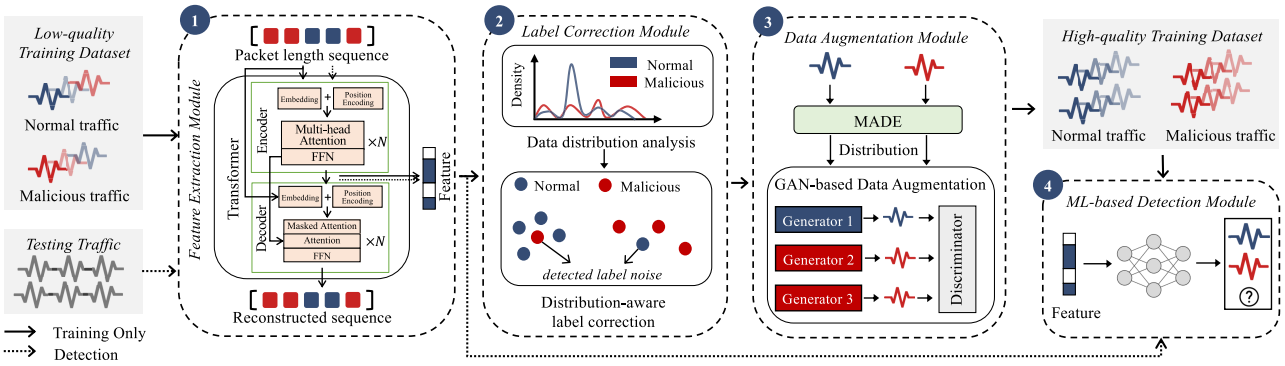


Fig. 1. The overview of our robust encrypted malicious traffic detection system RAPIER+.

III. Problem Statement

This paper aims to identify the encrypted malicious traffic generated by malware, with only a low-quality training set. Specifically, the amount of training data is limited, and the data is with non-negligible label noises. Formally, let (x_i, y_i) be an encrypted traffic data x_i and its true label $y_i \in \{0, 1\}$, where 0 or 1 represents a normal or a malicious one, respectively. The inputs of a detection system are a low-quality training set $\widetilde{D}_{\text{train}} = \{(x_i, \widetilde{y}_i)\}_{i=1}^{N}$ and a testing set $D_{\text{test}} = \{x^{\text{test}}\}$, where $N \ll |D_{\text{test}}|$ and $\widetilde{y}_i$ is a *noisy* label that may be inconsistent with y_i. Our goal is to accurately infer the label of x^{test} by using $\widetilde{D}_{\text{train}}$.

Our goal is to address the limitations of existing encrypted malicious traffic detection methods [2], [3], [26], [48], [49], [50], [68], [71], [74], [80], [84], [92], [101], [104], [105], [11], [52], [65], [94], [100], and develop a system that is applicable in more realistic scenarios. First, we consider two problems in the training data, namely, insufficient data and non-negligible label noises, whereas prior works only try to solve one problem. Second, our system does not require additional training data. We do not consider the scenario in which malicious traffic exhibits identical distributions as normal traffic over time. which is an extreme version of concept drift [54] and is better addressed by introducing more fine-grained features and re-collecting the entire training set.

IV. Detecting Encrypted Malicious Traffic

A. Overview

We design our system based on a key observation that the distribution of normal and malicious data is different. First, the normal traffic is relatively representative. [37], [59], [79]. Second, since malware are various in real networks, the malicious traffic is relatively manifold.

Thereby, given a noisy training set, we can infer the true labels of training samples located in the densest and sparsest parts of its data distribution and use them as a basis for correcting other labels. Moreover, the distribution of new malicious data may be closer to normal data over time since attackers are likely to imitate normal behaviors to avoid detection. Thus, we can infer the distribution of new malicious data and then synthesize new training data to improve the performance of ML-based detectors on new, unseen testing data.

RAPIER+ consists of three main components, including a feature extraction module, a label noise correction module, and a data augmentation module, as shown in Figure 1. The feature extraction module utilizes a Transformer model to convert encrypted traffic into feature vectors that capture the comprehensive behavior of the traffic, while mitigating the adverse effects of label noise in the low-quality training set and traffic distortion caused by dynamic network conditions. Next, the label noise correction module applies an auto-regressive generative model to accurately estimate the distribution of limited yet high-dimensional training data and relabel the training samples' labels based on their distribution characteristics. Then, with the label-corrected training set, the data augmentation module infers the possible distribution regions of new malicious data and applies an improved GAN model to synthesize new malicious training data located in these target regions. Meanwhile, to prevent the newly generated malicious data from increasing false alarms, this module synthesizes new normal training data that can maintain the decision boundary of the model. Finally, our system utilizes a DL-based detector built upon Co-teaching [32] and a Multi-layer Perceptron (MLP) that is trained on the synthesized and the label-corrected original training data. Here, we apply Co-teaching to eliminate the impacts of the small amount of remaining label noise in the training set. In the testing stage, the encrypted traffic is inspected by the built ML-based detector directly after being converted into feature vectors.

B. Feature Extraction

The feature extraction module converts raw encrypted traffic into feature vectors that represent the traffic's behaviors, which facilitates label correction and data augmentation afterward. Existing automatic traffic feature extraction methods [1], [16], [68], [72], [74] utilize supervised models to profile the behaviors of network traffic with various versions of encryption protocols. However, they are ill-suited because the non-negligible label noises in the low-quality training set will result in inaccurate feature selection. Even worse, the order of packets in encrypted traffic may change unpredictably due to dynamic network conditions [5], [13], [66], [92], such as varying hosts or different transmission paths. As a result, LSTM [36] and GRU models [12], which are widely used in existing traffic feature extraction methods, struggle to accurately profile the behavior of dynamically distorted encrypted traffic because of their high sensitivity to input data order. On the other hand, although less sensitive to input order, CNN models [44] can only capture partial behaviors between

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

closely adjacent packets due to their limited receptive fields, making them insufficient for learning comprehensive behaviors across all packets.

To accurately capture the comprehensive behavior of encrypted network traffic while mitigating the effects of label noise and traffic distortion, our feature extraction module adopts a Transformer [82] model. The rationale behind this design is two-fold. First, the Transformer model can be trained in an unsupervised manner, which inherently helps reduce the negative impact of label noise in the training set. Second, the Transformer can comprehensively and accurately capture the global and order-independent correlations between each pair of packets within encrypted traffic. Leveraging these fine-grained correlations as behavioral patterns of encrypted traffic allows for more effective differentiation between malicious and benign traffic.

In particular, our Transformer-based feature extraction module comprises an encoder, which extracts feature vectors from encrypted traffic to represent their behaviors, and a decoder, which reconstructs the original traffic using the feature vectors. We first divide raw encrypted traffic into network flows based on the five-tuple information and use the sequence of packet lengths of each flow as the input data for feature extraction [67]. We use $l = [l_1, l_2, \ldots, l_n]$ to denote the packet length sequence of n packets in a flow, where l_i is the length of the i-th packet. The encoder first embeds l using a learnable matrix $M \in \mathbb{R}^{L\times d}$, where L is the number of individual packet lengths and d is the embedding dimension. We denote the embedded input sequence as $v = [v_1, v_2, \ldots, v_n]$, where v_i is a d-dimensional vector corresponding to l_i and obtained by retrieving the l_i-th row of M. The encoder also adds positional encoding [82] to v to incorporate positional information. It then uses multiple stacked encoding layers to learn feature vectors from v, where each layer applies a multi-head self-attention mechanism to capture the correlations between each pair of packet lengths. Specifically, the first encoding layer utilizes three learnable matrices $W_Q \in \mathbb{R}^{d\times d_q}, W_K \in \mathbb{R}^{d\times d_q}, W_V \in \mathbb{R}^{d\times d_v}$ to project v into a query matrix $Q = vW_Q$, a key matrix $K = vW_K$, and a value matrix $V = vW_V$, respectively, for each head. The attention of v in each head is computed by:

$$\text{Att}(v) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{1}$$

Note that the attention of v contains the correlation information between each pair of packet lengths. Moreover, the encoding layer independently computes the attention for each head and concatenates the results to enrich the extracted correlation information. The concatenated attention $\text{MultiAtt}(v)$ is then processed by:

$$\begin{aligned} x &= \text{LayerNorm}\left(v + \text{MultiAtt}(v)\right), \\ \text{FFN}(x) &= \max(0, xW_1 + b_1)W_2 + b_2, \\ z &= \text{LayerNorm}\left(x + \text{FFN}(x)\right), \end{aligned} \tag{2}$$

where LayerNorm represents a layer-wise normalization operation, FFN represents a feed-forward fully connected layer, and $z = [z_1, z_2, \ldots, z_n]$ is the output of the first encoding layer. These operations aim to preserve the original packet length information in the output attention. Finally, z serves as the input to the second encoding layer, and so on. Each encoding layer shares the same design. The output of the final encoding layer is treated as the feature vector of the traffic flow corresponding to v.

The decoder comprises multiple stacked decoding layers and a reconstruction layer. Its input is the feature vector output by the encoder, and its output is the reconstructed packet length sequence. Specifically, each decoding layer first processes the input using a masked multi-head self-attention mechanism, where the mask ensures that the attention for the i-th data point is computed only based on the first to the (i-1)-th data points in the input sequence. This allows the Transformer model to learn to predict subsequent packet lengths based on previous ones. Next, each decoding layer computes cross-attention between the self-attention result and the feature vector, where the Query matrix is derived from the self-attention output, and the Key and Value matrices are mapped from the feature vector. This step leverages the feature vector to guide the reconstruction of the packet length sequence. Finally, each decoding layer performs layer normalization on the cross-attention result and further processes it through a feed-forward neural network (FFN), as shown in Eq. 2. Note that the output of one decoder layer serves as the input to the next. The reconstruction layer maps the output of the final decoding layer back to the packet length sequence using a linear transformation followed by a softmax function.

The feature extraction module is trained using an unsupervised method, *i.e.* optimizes the Transformer model based on the difference between the input and the reconstructed packet length sequences. This training method does not require the labels of training samples, and it allows the Transformer model to learn the intrinsic behaviors of encrypted traffic as feature vectors. Specifically, we use the cross-entropy loss function to measure the difference between the input and reconstructed packet length sequences, and apply the Adam optimizer for model optimization. The feature extraction module is first trained using $\widetilde{D}_{\text{train}}$. After training, the encoder is used to extract the feature vectors of the training traffic samples, which are then used as inputs to the label noise correction module. During the detection phase, the testing encrypted traffic is fed into the encoder to obtain feature vectors for detection.

The previous version of our work, RAPIER [67], employs a GRU-based autoencoder for feature extraction. However, due to its inherently sequential processing nature, it struggles to accurately characterize the behavior of distorted encrypted traffic caused by dynamic network conditions in real-world environments. By designing a novel Transformer-based feature extraction module, RAPIER+ overcomes this limitation.

C. Label Noise Correction

The label noise correction module corrects labels in the low-quality training set by identifying the different distribution between normal and malicious data. This approach is based on our observation that normal data tends to have a denser distribution than malicious data [67]. Therefore, we can estimate the distribution of the low-quality training data and then identify the samples that are located in the densest and sparsest regions of the distribution, as shown in Figure 2. These samples can then be labeled as normal and malicious, respectively. However, there are a number of challenges to achieving this goal. First, the extracted features

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

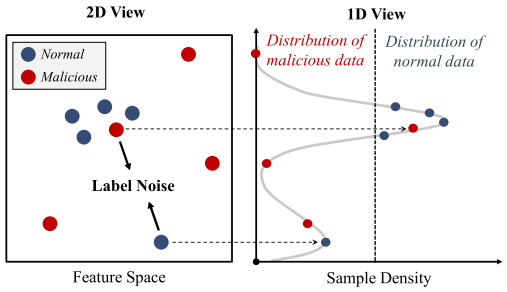


Fig. 2. The basic idea of our label noise correction module. The samples with high (*resp.* low) densities are more likely to be the true normal (*resp.* malicious) samples.

are of high dimensionality. Traditional statistical methods like kernel density estimation [81] cannot accurately estimate the distribution of high-dimensional data. Second, the difference between normal and malicious data distributions may not be significant, especially for the less frequent normal behaviors. It becomes difficult to infer the true labels of all training samples solely based on their distributions.

We develop our label noise correction module to overcome these issues. First, we estimate the distribution of training data based on a deep generative model, which can capture the probability distribution of observable high-dimensional variables [42]. According to the estimated distribution, we relabel part of the training samples that have the most significant distributions and then utilize them as a basis to infer the true labels of other training data by specially training a Transformer-based neural network.

Specifically, we leverage MADE [27], a highly efficient auto-regressive generative model, for distribution estimation. MADE models the distribution of input data by learning its probability density function. Assume that X is the input data set for MADE and $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ is an sample belonging to X. MADE will output a series of conditional probability densities $p(x_1)$, $p(x_2|x_1)$, $p(x_3|x_1x_2)$, $\ldots$, $p(x_d|x_1x_2x_3\ldots x_{d-1})$ for x. Then, the joint probability density of x can be computed as:

$$p(x) = \prod_{i=1}^{d} p(x_i|x_1\ldots x_{i-1}),$$
$$\text{with } p(x_i|x_1\ldots x_{i-1}) = p_{\mathcal{M}}(x_i|\zeta_i), \tag{3}$$

where $p_{\mathcal{M}}$ is a Gaussian mixture function with learnable parameters ζ_i. MADE learns the probability density function of X by maximizing the likelihood of each sample. The probability density of a sample approximately reflects a sample's position in the data distribution, *e.g.* in the densest or the sparsest region, which helps infer the sample's true label.

We select the set of training samples having normal labels (denoted as $\widetilde{w}_{\text{train}}$) from the original low-quality training set $\widetilde{D}_{\text{train}}$ and only use $\widetilde{w}_{\text{train}}$ as the input data set for MADE. The intuition behind this design is that the $\widetilde{w}_{\text{train}}$ usually contains less true malicious data than $\widetilde{D}_{\text{train}}$, and thus the strategy prevents MADE from assigning high-density values to true malicious data and effectively enlarges the separation between normal and malicious samples. In practice, benign traffic is typically collected from normal network activity on real hosts, making the proportion of malicious noise in the benign set consistent with the real-world malicious proportion, which is generally a minority [64]. This supports the validity of our intuition in real networks.[1]

After training MADE, we obtain the probability density values of each training sample in $\widetilde{D}_{\text{train}}$ and take the following steps to relabel some samples. First, we select a set $\widetilde{H}_{\text{train}}$ from $\widetilde{D}_{\text{train}}$, where the samples in $\widetilde{H}_{\text{train}}$ have higher probability density values than the ones outside the set. The size of $\widetilde{H}_{\text{train}}$ is set to $\alpha \cdot \left|\widetilde{D}_{\text{train}}\right|$ and $\alpha \in (0, 1)$ is a pre-defined parameter. Second, we measure its average Euclidean distance to others for each sample in $\widetilde{H}_{\text{train}}$ and select half of the samples with smaller distance values. The selected samples, denoted as N_s, are located in the densest regions of the data distribution and are similar to each other. Thereby, we can relabel them to be normal training data with high confidence. Third, for each sample in $\widetilde{D}_{\text{train}} - N_s$, we also compute its average Euclidean distance to all samples in N_s and select another set of samples M_s with larger distances. The samples in M_s are located in the sparse regions of the data distribution and are least similar to N_s. Thus, we relabel M_s to be malicious training data. Meanwhile, we set the size of M_s the same as N_s to prevent data imbalance in inferring the true labels of other samples.

Finally, with the label-corrected sets N_s and M_s, we employ a Transformer model trained in a supervised manner to infer the true labels of the remaining samples in $\widetilde{D}_{\text{train}}$. This model shares the same architecture as the feature extraction module. Specifically, we extract the packet length sequences of the samples in N_s and M_s to train the Transformer model. The training objective is to minimize the reconstruction error of the sequences extracted from N_s, while maximizing the reconstruction error of those from M_s. Let $\varepsilon(x)$ denote the reconstruction error of a sample x. The training objective of the supervised Transformer is defined as:

$$\mathcal{L}_{\text{label}} = \frac{1}{|N_s|}\sum_{x\in N_s}\varepsilon(x) - \frac{1}{|M_s|}\sum_{x\in M_s}\varepsilon(x). \tag{4}$$

We choose the Transformer model because it can comprehensively and accurately capture the behaviors of encrypted traffic, helping achieve good generalization capability even when trained on a limited number of samples in the corrected sets. Besides, we use the above loss instead of a classification loss because benign samples are dense and that can be well approximated by reconstruction, whereas malicious samples are sparse and thus naturally incur larger errors. This enables accurate inference of the labels of the remaining samples. In contrast, a classification loss would force the model to fit an overfit decision boundary. After training, we feed the packet length sequence of each remaining sample into the model. We then compute the reconstruction error for each sample and compare it to a threshold, which is set as the midpoint between the maximum and minimum reconstruction errors observed in the corrected sets. Samples with reconstruction error exceeding this threshold are labeled as malicious, while the others are considered normal.

Our previous version, RAPIER [67], uses traditional machine learning methods to infer the true labels of the remaining samples. These methods struggle to capture the

[1]We further discuss the extreme case where more than 50% of samples are mislabeled (*i.e.* worse than random labeling) in Section VI.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

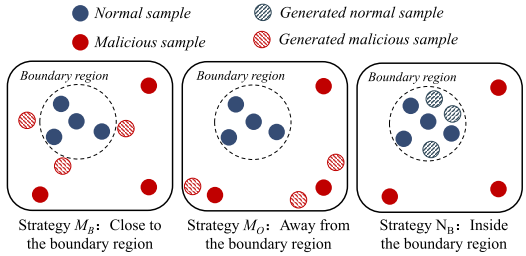


Fig. 3. Three data augmentation strategies: M_B and M_O are for augmenting malicious samples and N_B is for augmenting normal samples.

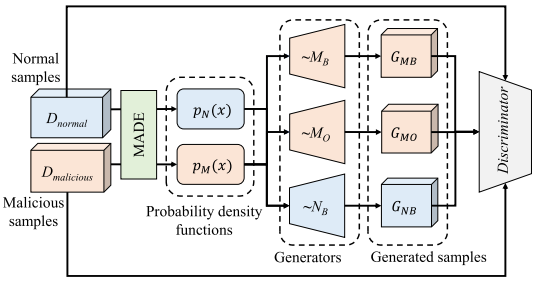


Fig. 4. The overview of our data augmentation module. We use three generators to generate samples (G_{MB}, G_{MO}, G_{NB}) based on three strategies (M_B, M_O, N_B), respectively. The $\sim M_B$ indicates the generator that fits the target distribution region of M_B, and the same goes for $\sim M_O$ and $\sim N_B$.

fine-grained behavior of encrypted traffic, resulting in limited generalization capability and inaccurate label inference.

D. Data Augmentation

The data augmentation module aims to synthesize new training data that can improve the detection performance on unseen testing data according to the label-corrected original training set D_{train}, which is non-trivial due to the following reasons: (i) We should ensure the diversity of the new training data. Otherwise, the machine learning model may still overfit the new training data, limiting its generalizability. (ii) The distribution of the testing data is likely to be inconsistent with the training data because malware is always evolving. As a result, the traditional data oversampling methods [8], [33], [34] and the popular GAN-based data synthesis approaches [9], [21], [30] are not applicable, since they only generate the data that follows the distribution of existing data. To address these issues, we predict possible distribution regions of new malicious data and sample from these target regions to synthesize new diverse training data.

Our data augmentation module augments training data by synthesizing both normal and malicious data according to data distributions. We argue that new malicious data may be located in two specific regions following the data distribution. First, considering that sophisticated attackers tend to imitate normal behaviors to evade detection, new malicious data may become more similar to normal data over time. Therefore, in distribution, some new malicious data may be located in the boundary region outside the normal data and yet very close to it (denoted as M_B). Second, since new attack methods are emerging, *e.g.* Zero-day attacks, new malicious data may be different from both normal and original malicious data, *i.e.* located in the region outside all original training data (denoted as M_O). To avoid data imbalance, we choose the boundary region inside the original normal data to guide the generation of normal data (denoted as N_B), which is essential for maintaining the decision boundary of the detection model. We visualize these strategies in Figure 3.

We first use the density functions to describe the target regions. In particular, we estimate the data distribution of D_{normal} and $D_{\text{malicious}}$ via MADE and use $p_N(\cdot)$ and $p_M(\cdot)$ to denote the corresponding probability density functions, respectively. The target regions corresponding to the three augmentation strategies can be represented as:

$$M_B = \{x | p_M(x) < \gamma, \quad \omega_1 \le p_N(x) < \omega_2\},$$
$$M_O = \{x | p_M(x) < \gamma, \quad 0 \le p_N(x) < \omega_1\},$$
$$N_B = \{x | p_M(x) < \gamma, \quad \omega_2 \le p_N(x) < \omega_3\}, \tag{5}$$

where $x \in \mathbb{R}^d$, ω_1, ω_2, ω_3, γ are pre-defined thresholds to control the sizes and locations of the target regions. In our implementation, we set $\omega_1 = 0.1$, $\omega_2 = 0.2$, and $\omega_3 = 0.3$, representing 10th, 20th, and 30th percentiles of the $p_N(x)$ of the benign set, respectively. We set these thresholds based on benign sample densities in the feature space. The lowest 10% capture sparse, scattered points far from the main cluster (M_O), 10%–20% lie at the cluster's outer edge but remain coherent (M_B), and 20%–30% are near the inner cluster boundary, not far enough to define a decision boundary (N_B).

Then, we develop an improved GAN [30] model for data augmentation, as shown in Figure 4. Unlike the vanilla GAN model approximating the distribution of the original training data, our model utilizes customized loss functions to learn the target data distributions. It consists of three *generator* networks and a *discriminator* network. Each *generator* is responsible for synthesizing new data that follows a target distribution, and the *discriminator* categorizes the newly generated data to perform adversarial training. We use X_G to denote the new data generated by one *generator*. The training goal of this *generator* is to minimize the Kullback-Leibler (KL) divergence between X_G and the distribution for each of M_B, N_B, M_O:

$$\begin{aligned}\mathcal{L}_{\text{KL}} = &-\mathcal{H}(X_G) \\ &+ \mathbb{E}_{x \in X_G, p_M(x) \ge \gamma} [\log p_M(x)] \\ &- \mathbb{E}_{x \in X_G, p_M(x) < \gamma, p_N(x) < \theta_1} [\log p_N(x)] \\ &+ \mathbb{E}_{x \in X_G, p_M(x) < \gamma, p_N(x) \ge \theta_2} [\log p_N(x)],\end{aligned} \tag{6}$$

where $\mathcal{H}$ is the entropy function that can be approximated by the pull-away term [103]. θ_1 and θ_2 will be replaced with the parameters relevant to this region, *e.g.* ω_1 and ω_2 for M_B. Note that the minimum $p_N(x)$ threshold of M_O is 0. This loss function allows our GAN to generate new samples in the area containing no existing samples and makes our data augmentation method superior to the vanilla GAN.

Since network traffic always follows specific protocol specifications, we need to prevent newly synthesized data from deviating from the original training data. Thereby, for the *generators* of M_B and N_B, we select the original training samples that already locate in the corresponding target region, *i.e.* $X_{\text{in}} = \{x | x \in D_{\text{train}}, p_M(x) < \gamma, \theta_1 \le p_N(x) < \theta_2\}$, and

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

minimize their difference with the new data generated by this *generator*. This goal can be defined as:

$$\mathcal{L}_f = \left\| \mathbb{E}_{x \in X_G} \left[\mathcal{D}_f(x) \right] - \mathbb{E}_{x \in X_{\text{in}}} \left[\mathcal{D}_f(x) \right] \right\|_2, \tag{7}$$

where $\mathbb{E}$ is the expectation function and $\mathcal{D}_f$ is the first layer of the *discriminator*. Then, the complete loss function for the *generators* of M_B and N_B is as follows:

$$\mathcal{L}_G = \mathcal{L}_{\text{KL}} + \mathcal{L}_f. \tag{8}$$

Besides, for the *generator* of M_O, its loss function only uses $\mathcal{L}_{\text{KL}}$ because the samples it synthesizes lie outside the regions of the original training data.

We denote the new data generated by three *generator* networks (*i.e.*, $\sim M_B$, $\sim M_O$ and $\sim N_B$) as G_{MB}, G_{MO} and G_{NB}, respectively. They are fed into the *discriminator* with D_{train}. Similar to the vanilla GAN, our *generator* generates new data that follows target distributions with high diversity. Our *discriminator* performs adversarial training to ensure the new data synthesized by the *generators* accurately lie in the target regions. Specifically, for $\sim M_B$ and $\sim N_B$, the *discriminator* learns to distinguish them between the original training data in their corresponding target regions, thereby increasing their probability of being generated within these regions. For $\sim M_O$, however, the *discriminator* learns to confuse it with the original training data, thereby increasing its probability of being generated outside the regions of all original training data. The loss function for our *discriminator* is defined as:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}} = & - \left\| \mathbb{E}_{x \in G_{MB}} \left[\mathcal{D}_f(x) \right] - \mathbb{E}_{x \in X_{MB}} \left[\mathcal{D}_f(x) \right] \right\|_2 \\ & - \left\| \mathbb{E}_{x \in G_{NB}} \left[\mathcal{D}_f(x) \right] - \mathbb{E}_{x \in X_{NB}} \left[\mathcal{D}_f(x) \right] \right\|_2 \\ & + \left\| \mathbb{E}_{x \in G_{MO}} \left[\mathcal{D}_f(x) \right] - \mathbb{E}_{x \in D_{\text{train}}} \left[\mathcal{D}_f(x) \right] \right\|_2, \end{aligned} \tag{9}$$

where X_{MB}, X_{NB} are parameter-specific instantiations of X_{in}, corresponding to $p_N(x)$ ranges of M_B, N_B, respectively.

Interpretability. In the real world, N_B may correspond to benign traffic occasionally affected by factors such as latency or packet loss, causing minor disorder or retransmissions within short segments of packet sequences. Meanwhile, M_B may represent malicious traffic that mimics long segments of benign flows while embedding packets carrying malicious functionality, making the traffic appear similar to benign samples in most features.

Our GAN model is trained with the label-corrected original data D_{train} by adversarial training. When the model converges, the new data synthesized by the *generator* networks will be regarded as the new training data. Besides, to avoid the model collapse problem [7] and further increase the diversity of synthesized data, our data augmentation module trains several independent GAN models and combines all their synthesized data as the new training data. Then, we can fully train the detection model based on the new training data and D_{train} to perform accurate detection. For comparison, the previous version of our work, RAPIER [67], does not employ an adversarial training-based *discriminator* design. As a result, it struggled to generate new samples that precisely fall within the target regions.

V. Evaluation

We evaluate the effectiveness of our detection system using public and real-world datasets. We also compare the performance of our work with representative encrypted malicious traffic detection methods.

TABLE I
The Parameter Settings of RAPIER+

Module	Parameter	Value	Description
Feature Extraction	n_{layer}	6	Number of Transformer layers
	n_{head}	8	Number of attention head

A. Experimental Setup

Datasets & Baselines. We use two public datasets for evaluation: CIRA-CIC-DoHBrw-2020 (DoHBrw) [60] and CSE-CIC-IDS2018 (IDS) [70], and use following baselines for comparisons: ETA [3], FS [50], DT [93], ODDS [37], SMOTE [8], and Co [32]. For comprehensive comparisons, we produce more baselines by combining existing baselines. The detailed description of the datasets and baselines is consistent with our prior work [67].

Implementation We implement RAPIER+ and all baselines by using Python 3.13.2, and the used libraries include NumPy 2.2.5, Pytorch 2.7.0, and Cuda 12.6. We run these models on a Linux server (6.5.0-41-generic) with Intel(R) Xeon(R) Gold 6258R CPU and NVIDIA GeForce RTX 4090 GPU. We list the parameters used individually in RAPIER+ in Table I. Other parameters are consistent with the prior work [67].

Metrics. We utilize three common metrics to evaluate the performance of our detection system, including precision, recall, and F1 score, which is consistent with the prior work.

B. Overall Detection Performance

We first evaluate the detection performance of our system and all baselines on the two public datasets, *i.e.* DoHBrw and IDS. Further, to evaluate RAPIER+ with an extreme scenario of malware evolution, we create a new dataset IDS/DoHBrw in the same way in our prior work.

We set three training sizes: 250, 500, and 1000. Each size represents the number of malicious and normal training samples randomly selected from T1 in each experiment. This random selection strategy ensures the diversity of the training samples. More importantly, for fair comparisons, we follow the label noise ratio settings used in Co [32] and DT [93]. In particular, the label noise ratio is defined as the proportion of mislabeled samples in all training samples, and there are six different noise ratios (20%, 25%, 30%, 35%, 40%, and 45%) for evaluations. The detailed process for each experiment setting is consistent with the prior work.

Impact of Noise Ratio. We show the F1 score of all methods under different noise ratios in Figure 5, and the training size is set to 500 (*i.e.* 500 normal and 500 malicious training samples, 1000 samples in total). It can be seen that our system outperforms all baselines with significant margins in all settings. For instance, when the noise ratio is as high as 45%, our system still achieves the best F1 scores of 0.78, 0.84, and 0.87 on the DoHBrw, IDS, and IDS/DoHBrw datasets, respectively, showing average improvements of 358.5%, 314.0%, and 221.1% over the baselines. Note that the average improvement

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

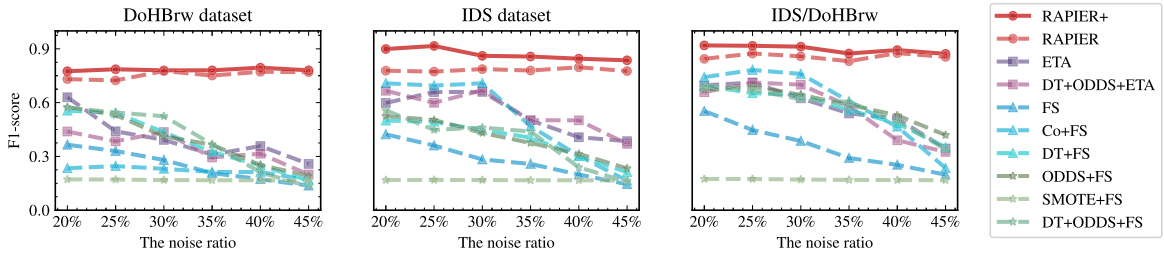


Fig. 5. The F1 score of all methods under different noise ratios on three evaluation datasets.

TABLE II

THE F1 SCORE OF ALL METHODS UNDER DIFFERENT TRAINING SIZES WITH THE NOISE RATIO OF 30%. THE F1 SCORE IS SHOWN IN THE FORM OF "AVG ± STD", WHERE AVG IS THE AVERAGE F1 SCORE AND STD IS THE STANDARD DEVIATION

Method		FS	Co+FS	DT+FS	ODDS+FS	SMOTE+FS	DT+ODDS+FS	ETA	DT+ODDS+ETA	RAPIER	RAPIER+
Training Size on DoHBrw	250	.18 ± .02	.21 ± .06	.41 ± .02	.42 ± .05	.17 ± .00	.34 ± .10	.56 ± .27	.44 ± .21	.71 ± .02	**.73** ± .02
	500	.28 ± .04	.23 ± .06	.44 ± .03	.42 ± .07	.17 ± .00	.52 ± .04	.39 ± .24	.44 ± .23	.78 ± .02	**.78** ± .03
	1000	.30 ± .03	.27 ± .05	.54 ± .02	.57 ± .06	.17 ± .00	.58 ± .05	.40 ± .23	.29 ± .07	.78 ± .02	**.81** ± .02
Training Size on IDS	250	.26 ± .05	.66 ± .19	.34 ± .01	.46 ± .04	.17 ± .00	.38 ± .03	.57 ± .10	.53 ± .15	.75 ± .04	**.86** ± .05
	500	.28 ± .04	.71 ± .06	.45 ± .04	.43 ± .06	.17 ± .00	.46 ± .08	.66 ± .27	.67 ± .24	.79 ± .02	**.86** ± .11
	1000	.35 ± .01	.73 ± .08	.51 ± .03	.51 ± .08	.17 ± .00	.54 ± .05	.66 ± .27	.69 ± .23	.77 ± .02	**.85** ± .10
Training Size on IDS/DoHBrw	250	.31 ± .13	.66 ± .19	.51 ± .10	.58 ± .12	.17 ± .01	.53 ± .11	.59 ± .12	.57 ± .15	.82 ± .09	**.89** ± .17
	500	.39 ± .14	.76 ± .06	.64 ± .13	.64 ± .15	.17 ± .00	.62 ± .14	.62 ± .27	.70 ± .27	.86 ± .05	**.88** ± .09
	1000	.38 ± .13	.73 ± .08	.68 ± .09	.69 ± .13	.17 ± .00	.72 ± .13	.69 ± .24	.73 ± .19	.81 ± .07	**.84** ± .11

is the mean value of the improvement of our system to each baseline. Besides, we observe that the varying noise ratio only slightly affects our detection performance. In particular, when the noise ratio increases from 20% to 45%, the F1 score fluctuations of our system on the DoHBrw, IDS, and IDS/DoHBrw datasets are only about 0.02, 0.08, and 0.05, respectively. By contrast, the average F1 score fluctuations of all baselines except SMOTE are above 0.3. The F1 score of SMOTE keeps lower than 0.2, so we do not compute its performance fluctuations. These results indicate that when facing highly insufficient training data, our system is much more robust to label noise than all baselines. This is because our system effectively corrects the incorrectly labeled samples based on their distinctive distribution characteristics, which is less relevant to the number of correctly labeled samples. Specifically, the number of label noises after label correction is stable as the original noise ratio changes, and the detailed performances and explanation of the label-correction module are discussed in § V-D.

We also compare RAPIER+ with our prior work RAPIER and show the results in Figure 5. RAPIER+ gain higher performances on all settings, achieving average improvements of 3.9%, 11.2%, and 4.9% on the three datasets, respectively. The performance improvements of our new work are due to three key factors. First, the Transformer-based feature extraction module profiles more comprehensive and accurate behaviors of encrypted traffic into feature vectors. Second, the label correction module uses a transformer model to infer the true labels of low-quality training traffic more accurately. Further, the data augmentation module employs an adversarial training-based discriminator to generate new samples that accurately locate in target distribution regions.

Impact of Training Size. We show the F1 score of all methods under different training sizes in Table II. The noise ratio is set to 30%. Our system outperforms baselines in all settings. When the training size is as small as 250, our system achieves the F1 scores of 0.73, 0.86, and 0.89 on the DoHBrw, IDS, and IDS/DoHBrw datasets, respectively, with average improvements of 155.7%, 143.3%, and 118.5% to all baselines. It shows that even when plenty of training samples are incorrectly labeled, our system can synthesize new, useful training data for improving detection performance. In contrast, the baselines that solely utilize existing data augmentation methods, *e.g.* ODDS+FS and SMOTE+FS, cannot deal with the negative impacts of label noise. Moreover, our system remains stable under different training sizes. The F1 score fluctuations of our system on the DoHBrw, IDS, and IDS/DoHBrw datasets are 0.08, 0.01, and 0.05, respectively. It shows that our data augmentation strategies are applicable to different low-quality training sets. In contrast, the baselines' performance shows significant degradations as the training size varies, *e.g.* the F1 scores of DT+ODDS+FS are reduced by 0.24, 0.16, and 0.19 on the three datasets when the training size decreases from 1000 to 250. We also notice that all methods can achieve higher detection performance on the IDS/DoHBrw dataset than on the IDS dataset since the malicious samples in the DoHBrw dataset are less similar to the normal samples in the IDS dataset. In this case, our system still outperforms all baselines by a significant margin. This is because the new training data synthesized by our system is more representative of the malware samples in the DoHBrw dataset, allowing our system to learn more about the characteristics of malicious traffic. Note that, compared to RAPIER, RAPIER+ achieves an average improvement of 2.2%, 11.3%, and 4.9% on the three datasets, respectively, which is also due to the effectiveness of our new designs.

We supplement the detection performance of all methods with each training size and noise ratio setting in Appendix A. Our system achieves the best performance in most cases. The reason behind this is that our system accurately corrects label

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

TABLE III

THE F1 SCORES OF RAPIER+ WITH SMALLER RATIOS OF MALICIOUS TO NORMAL SAMPLES. THE TRAINING SIZE IS 500 AND THE NOISE RATIO IS 30%

Dataset		DoHBrw	IDS	IDS/DoHBrw
Malicious Ratio (malicious : normal)	1:10	.78 ± .03	.86 ± .11	.88 ± .09
	1:20	.69 ± .02	.84 ± .13	.84 ± .11
	1:30	.60 ± .02	.81 ± .17	.83 ± .12
	1:50	.48 ± .03	.75 ± .22	.82 ± .18

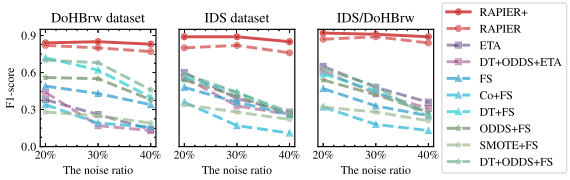


Fig. 6. The F1 scores under realistic label noise settings.

noises within low-quality training data and synthesizes new training data that can represent unseen malware samples.

Impact of smaller malicious-to-normal ratios. We further measure the performance of our framework with smaller ratios of malicious to normal samples in the testing sets. We select three ratios: 1:20, 1:30, and 1:50. The training size is 500, and the noise ratio is 30%. The results are shown in Table III. We can see that when the ratio increases from 1:10 to 1:30, the F1 scores of our framework only decrease approximately by 5.8% and 5.7% on the IDS and IDS/DoHBrw datasets, respectively. Even when the ratio reaches 1:50, our framework still achieves about 0.48, 0.75, and 0.82 F1 scores on the DoHBrw, IDS, and IDS/DoHBrw datasets, respectively. This demonstrates that our framework can effectively detect encrypted malicious traffic in real-world networks where malicious traffic is rare. Additionally, the F1 score decreases due to low precision scores. For example, the precision on the DoHBrw dataset decreases from 0.68 to 0.47 when the ratio rises from 1:10 to 1:30, indicating that more false alarms are reported. We will discuss this issue in § VI.

C. Various Settings of Label Noises

To evaluate the performance of our system under more realistic label noise settings, we create label noise based on two common scenarios where malicious and normal encrypted traffic can be mislabeled. The process for building such a noisy dataset is consistent with the prior work. We set the training size to 500 and the noise ratio to 20%, 30%, and 40%. For each training size and noise ratio setting, we choose one malware type at a time and calculate the average detection performance of each method under all malware types. As shown in Figure 6, our system outperforms all baseline models. Specifically, our system achieves average F1 scores (among all settings) of 0.84, 0.88, and 0.91 on the DoHBrw, IDS, and IDS/DoHBrw datasets, respectively, achieving average improvements of 182.6%, 181.6%, and 177.7% over baselines and 5.5%, 10.5%, and 4.6% over RAPIER. This indicates that RAPIER+ can perform robust encrypted malicious network detection when trained with the low-quality training data generated in realistic circumstances. We notice that most baselines' performance decreases compared to their results

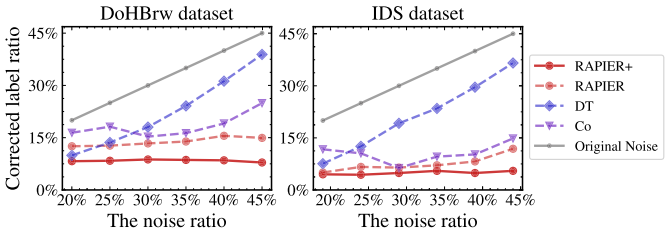


Fig. 7. The corrected and remaining noise ratios under different original noise ratios.

in § V-B (when samples are randomly selected to be label noises). For instance, the F1 score of ETA declines 30% on average. In contrast, RAPIER+ maintains similar detection accuracy, with F1 score fluctuations of less than 0.04.

D. Evaluating Individual Components

Feature Extraction. We first compare our feature extraction module with two representative feature extraction methods ETA [2] and CICFlowMeter [43]. We replace our feature extraction module with these two methods while keeping our label noise correction and data augmentation modules.

The F1 score reduces to 0 when we use these two new feature extraction methods in all settings, and we inspect the results of the two modules. CICFlowMeter has recall scores of 0, meaning that it is unable to identify encrypted malicious traffic. This is likely because the coarse-grained statistical features extracted by CICFlowMeter are not able to distinguish between encrypted malicious and normal traffic. Besides, ETA extracts feature vectors of much higher dimensions (386) than our system (32). This makes it difficult for the deep generative model (MADE) to estimate the distribution of such high-dimensional data. As a result, the remaining components of our system, depending on the data distribution, cannot function properly. In contrast, we extract more important sequential features that represent fine-grained behaviors, where the extracted feature vectors are of lower dimensions, facilitating accurate distribution estimation performed by the MADE model.

Label Correction. Now we compare our label correction module with two SOTA label noise correction methods DT and Co. The evaluation method is the same as the prior work.

The label noise correction results are shown in Figure 7. Our module can reduce the original noise ratio to lower levels. It can reduce the noise ratio to 8.40% and 4.94% on the DoHBrw and IDS datasets, respectively. Our module is also stable at correcting noise. The standard deviations of the remaining noise ratios corresponding to our module are only 0.28% and 0.44% on the DoHBrw and IDS datasets, respectively. In contrast, the noise correction performance of the other two methods degrades as the original noise ratio increases. When the original noise ratio is 45%, the proportion of noises corrected by RAPIER+ is 82.5% on the DoHBrw dataset, which is 6.08 times higher than DT and 1.85 times higher than Co. Our noises correction proportion on the IDS dataset is 87.78%, which is 4.69 times higher than DT and is 1.31 times higher than Co. Moreover, RAPIER+ improves RAPIER's reduction proportion by 33.0% and 9.6% on the two datasets, respectively. We notice that the performance of all methods on the IDS dataset is inferior to that on the

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
THE DETECTION PERFORMANCE AFTER DATA AUGMENTATION. P, R, AND F REPRESENT PRECISION, RECALL, AND F1 SCORE

Dataset	Metrics	NONE	ODDS	SMOTE	GAN	RAPIER	RAPIER+
DoHBrw	P	.70 ± .09	.67 ± .07	.64 ± .05	.70 ± .09	.76 ± .04	**.79** ± .02
	R	.92 ± .06	.95 ± .02	.95 ± .01	.91 ± .06	.94 ± .02	**.95** ± .02
	F	.79 ± .05	.78 ± .04	.77 ± .03	.78 ± .04	.84 ± .02	**.86** ± .01
IDS	P	.95 ± .06	.97 ± .02	.88 ± .24	.95 ± .08	.98 ± .00	**.98** ± .00
	R	.93 ± .04	.94 ± .02	.96 ± .02	.94 ± .01	.95 ± .01	**.98** ± .00
	F	.94 ± .04	.96 ± .01	.88 ± .21	.94 ± .05	.97 ± .00	**.98** ± .00
IDS/DoHBrw	P	.94 ± .08	.92 ± .21	.86 ± .29	.97 ± .02	.99 ± .00	**.99** ± .00
	R	.94 ± .02	.95 ± .02	.96 ± .03	.93 ± .04	.96 ± .01	**.96** ± .00
	F	.94 ± .04	.91 ± .19	.86 ± .26	.95 ± .02	.97 ± .00	**.98** ± .00

DoHBrw dataset, mainly due to the difference between the two datasets. In summary, our label correction module can effectively correct the incorrect labels under different noise rates, and its performance outperforms the SOTA methods.

Data Augmentation. We compare our data augmentation module with NONE (refers to perform detection without data augmentation), ODDS [37], SMOTE [8], and the vanilla GAN [30] model. The evaluation method is the same as the prior work.

We show the detection performance when the training size is 500 in Table IV. Our data augmentation module can achieve the highest F1 score on each dataset, namely, 0.86, 0.98, and 0.98 on the DoHBrw, IDS, and IDS/DoHBrw datasets, respectively. It achieves average improvements of 8.7%, 4.6%, and 6.0% over other data augmentation methods. Compared with the no augmentation strategy, our module improves both the precision and recall scores. For example, on the IDS dataset,RAPIER+ improves the recall from 0.93 to about 0.98 and the precision from 0.95 to about 0.98. This means our module can identify more unseen malicious testing samples without generating more false alarms. We notice that, in some cases, the detection performance of ODDS, GAN, and SMOTE is worse than that of the no augmentation strategy. This is because the new training samples synthesized by these baselines are not effective at improving the precision or recall scores. We also observe that a small number of normal samples exhibiting infrequent normal behaviors can lead to lower precision in some cases. This is likely because these samples are located outside dense distribution regions of the normal data and are falsely identified as malicious. Besides, the benefits of the augmentation method vary across different datasets due to the different packet types present in each dataset. For example, the IDS dataset generated by hundreds of hosts contains a wider variety of packet types than the DoH dataset, which mainly contains DNS requests. The increased diversity of packet types can lead to more false alarms and lower precision.

We supplement the experiment results with the training size of 250 and 1000 in Appendix B, and our module also achieves the best F1 score in these settings. In summary, our data augmentation module can effectively improve the detection capability of ML models built upon limited training data against evolving malware samples.

E. Parameter Deep Dive

Filtering Ratio α. We evaluate the impacts of different filtering ratios α values. Recall that α controls the proportion of high-confidence normal samples filtered at the first step of label correction. We set five different α values (0.40, 0.45, 0.50, 0.55, and 0.60). Then, we perform label correction experiments with different noise ratios. Specifically, we set the training size to 500 and make the noise ratio in the range of 20% to 45%. Figure 8 shows the remaining noise ratio results. Our label correction module can always achieve low remaining noise ratios (*i.e.* good label correction results) under different α settings. In particular, the average remaining noise ratios on the DoHBrw and IDS datasets are 7.39% and 4.39%, respectively. Moreover, the variance of the results regarding different α settings is relatively low (0.66% and 0.35% on the DoHBrw and the IDS dataset, respectively), demonstrating the stability of our label correction module.

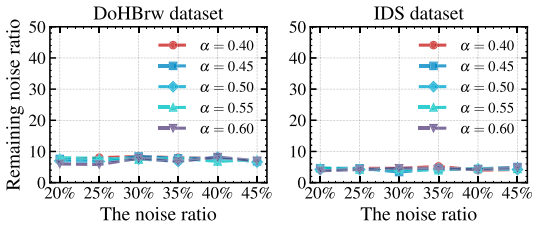


Fig. 8. Remaining noise ratios of various filtering ratios α.

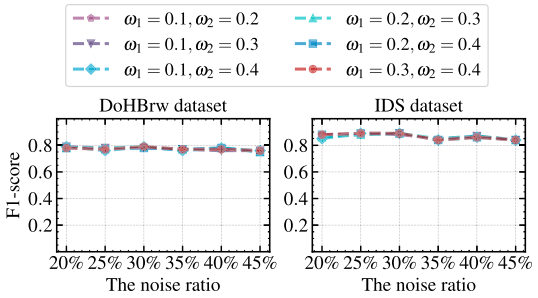


Fig. 9. The F1 score of different thresholds ω_1, ω_2.

Thresholds ω_1, ω_2 and ω_3. We also evaluate the impacts of different thresholds ω_1, ω_2, and ω_3 values with the same way in the prior work. In the experiments, the training size is also set to 500, and the noise ratio is in the range of 20% to 45%. We show the F1 scores of our system in Figure 9. It can be seen that our system achieves relatively stable detection performance when these thresholds vary. For instance, when the noise ratio is 30%, the standard deviations of F1 scores corresponding to different ω_1 and ω_2 combinations are only 0.005 and 0.004 on the DoHBrw and IDS datasets, respectively.

Number of GAN models η. We evaluate the impacts of different η values. When the noise ratio is 30%, the standard deviations of F1 scores corresponding to different η values on the two datasets are only 0.016 and 0.017. Also, the detection performance improves slightly as η increases. In particular, when η increases from 3 to 7, the average F1 scores on the DoHBrw and IDS datasets improve by 2.0% and 2.5%, respectively. It validates that more independent GAN models can enhance the diversity of the generated training samples, leading to better detection performance. Thus, we can choose either a large η value for better detection performance or a moderate η value to balance the detection performance and

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

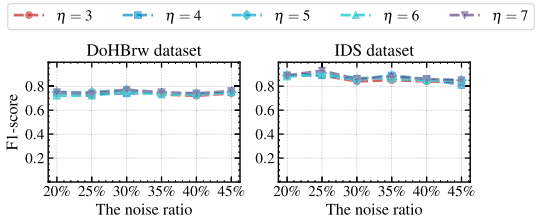


Fig. 10. The F1 score of different numbers of GAN models η.

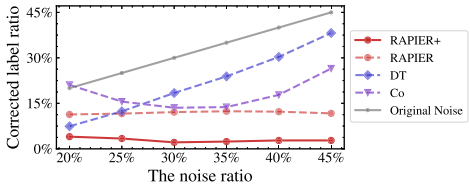


Fig. 11. The corrected and remaining noise ratios under different original noise ratios on the real-world dataset.

TABLE V

THE DETECTION PERFORMANCE AFTER DATA AUGMENTATION. P, R, AND F REPRESENT PRECISION, RECALL, AND F1 SCORE

Training size	Metrics	NONE	ODDS	SMOTE	GAN	RAPIER	RAPIER+
250	P	.63 ± .20	.73 ± .14	.69 ± .15	.55 ± .23	.76 ± .06	**.78** ± .03
	R	.83 ± .24	.95 ± .08	.98 ± .01	.77 ± .28	.94 ± .09	**.99** ± .00
	F	.70 ± .21	.81 ± .14	.80 ± .14	.59 ± .25	.84 ± .05	**.87** ± .02
500	P	.65 ± .20	.72 ± .14	.63 ± .21	.65 ± .18	.79 ± .02	**.83** ± .03
	R	.90 ± .22	.95 ± .09	.98 ± .03	.90 ± .22	.99 ± .01	**.99** ± .00
	F	.73 ± .22	.80 ± .15	.74 ± .21	.74 ± .20	.88 ± .01	**.90** ± .01
1000	P	.73 ± .12	.67 ± .23	.52 ± .29	.72 ± .12	.80 ± .04	**.85** ± .02
	R	.96 ± .04	.97 ± .08	.99 ± .00	.95 ± .05	.99 ± .01	**.99** ± .00
	F	.82 ± .09	.76 ± .22	.62 ± .30	.81 ± .09	.88 ± .02	**.91** ± .01

training overhead. The overall performance of RAPIER+ is not sensitive to parameter choices. Its good detection capability is attributed to our design, rather than specific parameters.

F. Real-World Experiments

We also use the same real-world dataset as the prior work to evaluate RAPIER+ performance in real-world scenarios. Specifically, the dataset is collected by a network security enterprise, including over 2.9M benign and 790K malicious encrypted traffic flows with timestamps ranging from Nov. 2017 to Feb. 2021.

We supplement the detection performance on the real-world dataset under all settings in Appendix C. Overall, our system outperforms all baselines with non-trivial margins in nearly all cases. When the training size is 500, RAPIER+ achieves the best F1 score of about 0.81, and also, RAPIER+ improves baselines by 288.7% and improves RAPIER by 2.6% on average on all noise ratio settings. We also evaluate our label noise correction module on this real-world dataset and show the results in Figure 11. Specifically, our label noise correction module achieves a steadily low remaining noise ratio (less than 4.0%) given different noise ratios in the training data, and the peak ratio for noise removal is 91%. Besides, we further evaluate our data augmentation module on this real-world dataset, and the results are shown in Table V. Compared with other augmentation approaches, the new training data synthesized by our module brings in the largest detection performance improvement, especially in the recall. Specifically, our data augmentation module outperforms other augmentation approaches by 19.6% precision and 8.4% recall score (on average). It shows our data augmentation module enables ML-based detectors to detect more realistic malicious samples unseen in the training data while reducing false alarms.

VI. DISCUSSION

Extreme Label Noises. The performance of our system will degrade inevitably when more than 50% of the data is mislabeled. However, benefiting from our dedicated design, our system is still more robust than other baselines in such extreme cases. For instance, on the IDS dataset, when the training size is 500, and the label noise ratio is set to 60%, 75%, and 90%, the F1 score of our system still remains at about 0.801, 0.697, and 0.494, respectively. Meanwhile, the performance of DT+ODDS+FS and DT+ODDS+ETA dramatically reduces to about (0.009, 0.010), (0.016, 0.017), and (0.003, 0.000), respectively. Besides, such extreme cases are rare in practice because we can pre-process the collected traffic data to reduce the label noise ratio to a reasonable level, *e.g.* less than 50%. For instance, we can filter the collected malicious encrypted traffic through the Alexa Top list and only preserve the normal encrypted traffic communicated with well-known benign servers, to reduce the incorrectly labeled traffic mixed in.

Training and Detecting Overhead. Typically, deep learning models involve large training time requirements, yet it is not a critical concern for our system. Since the low-quality training set for our system is usually of limited size, the deep learning models used by our system can be trained fast. For instance, when the number of training samples is 500, the feature extraction module, the label noise correction module, and the data augmentation module consume about 25.3, 8.21, and 137 seconds, respectively. Besides, some components in our system, *e.g.* the ensemble classifiers for label noise correction and the multiple GAN models for data augmentation, can be trained in parallel to boost efficiency. Even if the training size becomes larger, we can enhance our system with the recent efficiency improvement techniques, like over-specification [53] and dropout training [77]. After training, our system supports low-latency and high-throughput inference, with a latency of 2ms and a throughput of over $42,000$ network flows per second, making it suitable for real-time deployment.

Performance under Long-term Deployment. We observe that, over the long term, the deployment of our framework will lead to a gradual increase in both the diversity and the volume of normal traffic. This will result in more normal testing traffic exhibiting unseen normal behaviors. These new normal traffic data may be detected as malicious. A simple yet effective solution is to monitor key intermediate signals (e.g., reconstruction error) to detect concept drift and re-collect data and re-train the system after drift being detected. As we have evaluated, the training overhead of our system is small (25.3, 8.21, and 137 seconds on our three modules, respectively), which is quite efficient. We can also cluster these detected malicious samples based on their temporal intervals and filter out the isolated false alarms [25]. Besides, the fresh, diverse normal data collected under long-term deployment may affect

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
THE F1 SCORE OF EACH METHOD ON THE DOHBRW DATASET

Training size	Noise Ratio	FS	Co+FS	DT+FS	ODDS+FS	SMOTE+FS	DT+ODDS+FS	ETA	DT+ODDS+ETA	RAPIER	RAPIER+
250	20%	.24 ± .07	.24 ± .04	.53 ± .02	.52 ± .03	.16 ± .00	.30 ± .17	.62 ± .26	.37 ± .07	.74 ± .03	.71 ± .02
	25%	.20 ± .03	.20 ± .05	.48 ± .02	.47 ± .02	.16 ± .00	.52 ± .07	.53 ± .28	.37 ± .09	.83 ± .04	.67 ± .05
	30%	.18 ± .01	.20 ± .05	.41 ± .01	.41 ± .04	.16 ± .00	.33 ± .10	.56 ± .27	.44 ± .21	.69 ± .01	**.69** ± .04
	35%	.16 ± .01	.20 ± .04	.35 ± .02	.32 ± .02	.16 ± .00	.25 ± .08	.41 ± .23	.29 ± .08	.68 ± .01	**.68** ± .05
	40%	.15 ± .00	.18 ± .04	.27 ± .02	.27 ± .02	.16 ± .00	.30 ± .02	.47 ± .21	.26 ± .06	.54 ± .06	**.73** ± .05
	45%	.14 ± .00	.15 ± .07	.19 ± .00	.18 ± .02	.16 ± .00	.17 ± .00	.32 ± .25	.26 ± .22	.59 ± .03	**.72** ± .02
1000	20%	.44 ± .01	.20 ± .09	.66 ± .01	.63 ± .06	.17 ± .00	.57 ± .15	.44 ± .20	.36 ± .17	.58 ± .03	**.69** ± .02
	25%	.35 ± .01	.28 ± .06	.62 ± .02	.64 ± .02	.17 ± .00	.62 ± .04	.55 ± .24	.34 ± .03	.58 ± .01	**.73** ± .07
	30%	.29 ± .02	.27 ± .05	.54 ± .02	.56 ± .05	.16 ± .00	.57 ± .05	.39 ± .22	.29 ± .06	.64 ± .01	**.72** ± .07
	35%	.25 ± .03	.26 ± .04	.38 ± .05	.46 ± .02	.16 ± .00	.49 ± .04	.57 ± .32	.28 ± .11	.62 ± .02	**.71** ± .08
	40%	.20 ± .01	.22 ± .06	.30 ± .01	.32 ± .02	.16 ± .00	.28 ± .09	.41 ± .30	.28 ± .14	.62 ± .02	**.72** ± .08
	45%	.15 ± .00	.24 ± .05	.20 ± .02	.18 ± .05	.16 ± .00	.20 ± .02	.46 ± .31	.30 ± .13	.65 ± .02	**.65** ± .03

TABLE VII
THE F1 SCORE OF EACH METHOD ON THE IDS DATASET

Training size	Noise Ratio	FS	Co+FS	DT+FS	ODDS+FS	SMOTE+FS	DT+ODDS+FS	ETA	DT+ODDS+ETA	RAPIER	RAPIER+
250	20%	.35 ± .02	.73 ± .10	.44 ± .02	.45 ± .05	.16 ± .00	.48 ± .05	.70 ± .16	.67 ± .13	.79 ± .03	**.92** ± .05
	25%	.28 ± .03	.70 ± .10	.44 ± .01	.42 ± .05	.16 ± .00	.46 ± .06	.60 ± .24	.61 ± .20	.81 ± .01	**.89** ± .06
	30%	.26 ± .04	.65 ± .18	.06 ± .00	.45 ± .04	.16 ± .00	.15 ± .00	.57 ± .10	.52 ± .14	.75 ± .03	**.86** ± .05
	35%	.19 ± .02	.61 ± .17	.34 ± .01	.34 ± .05	.16 ± .00	.38 ± .02	.39 ± .22	.41 ± .23	.80 ± .01	**.90** ± .07
	40%	.15 ± .01	.34 ± .15	.25 ± .02	.24 ± .02	.16 ± .00	.21 ± .04	.38 ± .12	.40 ± .15	.79 ± .02	**.84** ± .09
	45%	.13 ± .01	.25 ± .19	.00 ± .00	.19 ± .02	.16 ± .00	.08 ± .07	.26 ± .15	.29 ± .13	.80 ± .01	**.88** ± .09
1000	20%	.49 ± .04	.77 ± .04	.59 ± .04	.60 ± .04	.17 ± .00	.62 ± .02	.83 ± .14	.83 ± .13	.77 ± .00	**.79** ± .11
	25%	.39 ± .02	.73 ± .05	.56 ± .06	.61 ± .05	.16 ± .00	.53 ± .07	.78 ± .23	.75 ± .27	.78 ± .01	**.80** ± .14
	30%	.35 ± .00	.72 ± .07	.50 ± .03	.50 ± .08	.16 ± .00	.54 ± .04	.65 ± .27	.68 ± .23	.77 ± .01	**.85** ± .10
	35%	.29 ± .00	.67 ± .09	.45 ± .02	.43 ± .07	.16 ± .00	.51 ± .12	.61 ± .31	.63 ± .31	.75 ± .03	**.86** ± .12
	40%	.23 ± .00	.63 ± .12	.37 ± .01	.34 ± .06	.16 ± .00	.39 ± .01	.45 ± .18	.42 ± .20	.77 ± .01	**.86** ± .09
	45%	.16 ± .01	.39 ± .13	.22 ± .01	.25 ± .04	.16 ± .00	.26 ± .02	.40 ± .22	.38 ± .21	.78 ± .00	**.81** ± .13

TABLE VIII
THE F1 SCORE OF EACH METHOD ON THE IDS/DOHBRW DATASET

Training size	Noise Ratio	FS	Co+FS	DT+FS	ODDS+FS	SMOTE+FS	DT+ODDS+FS	ETA	DT+ODDS+ETA	RAPIER	RAPIER+
250	20%	.33 ± .03	.73 ± .10	.43 ± .01	.53 ± .03	.16 ± .00	.47 ± .02	.70 ± .23	.62 ± .15	.81 ± .07	**.96** ± .04
	25%	.19 ± .01	.70 ± .10	.42 ± .03	.40 ± .04	.16 ± .00	.48 ± .04	.62 ± .19	.57 ± .20	.81 ± .09	**.94** ± .06
	30%	.16 ± .00	.65 ± .18	.08 ± .04	.42 ± .03	.16 ± .00	.15 ± .00	.58 ± .11	.57 ± .15	.81 ± .08	**.89** ± .17
	35%	.16 ± .01	.61 ± .17	.34 ± .02	.36 ± .01	.16 ± .00	.39 ± .02	.43 ± .22	.34 ± .14	.81 ± .10	**.89** ± .17
	40%	.13 ± .01	.34 ± .15	.16 ± .02	.24 ± .02	.16 ± .00	.17 ± .03	.37 ± .16	.38 ± .12	.81 ± .08	**.91** ± .09
	45%	.08 ± .00	.25 ± .19	.00 ± .00	.11 ± .03	.16 ± .00	.05 ± .05	.30 ± .21	.31 ± .17	.82 ± .07	**.87** ± .11
1000	20%	.43 ± .03	.77 ± .04	.61 ± .03	.59 ± .04	.17 ± .00	.61 ± .06	.87 ± .14	.79 ± .14	.77 ± .07	**.87** ± .12
	25%	.28 ± .03	.73 ± .05	.59 ± .02	.56 ± .09	.16 ± .00	.57 ± .09	.80 ± .24	.79 ± .17	.83 ± .04	**.91** ± .06
	30%	.23 ± .00	.72 ± .07	.51 ± .03	.49 ± .04	.16 ± .00	.52 ± .08	.68 ± .24	.72 ± .19	.75 ± .09	**.86** ± .08
	35%	.19 ± .00	.67 ± .09	.45 ± .04	.44 ± .07	.16 ± .00	.39 ± .14	.62 ± .27	.57 ± .28	.81 ± .04	**.88** ± .13
	40%	.13 ± .00	.63 ± .12	.33 ± .03	.36 ± .06	.16 ± .00	.34 ± .02	.48 ± .20	.43 ± .18	.79 ± .08	**.90** ± .08
	45%	.11 ± .00	.39 ± .13	.22 ± .03	.19 ± .03	.16 ± .00	.27 ± .02	.30 ± .16	.37 ± .21	.80 ± .06	**.85** ± .11

the effectiveness of our label correction module. To handle this issue, we can pre-process the normal training data via clustering, *i.e.* aggregating the normal data sharing similar behavior into the same training set, and then correct the label noises in each set individually. We leave detailed exploration in this regard to future work.

Evading Detection. Sophisticated attackers may try to bypass our system. One common strategy is generating adversarial examples by adding well-crafted perturbations in the encrypted network traffic of malware. Our system has built-in designs to handle adversarial examples. In particular, unlike traditional deep learning models designed for end-to-end tasks, our system uses deep generative models and GAN models to improve the quality of training data *before* training the subsequent ML-based detector. This creates an extra barrier for the evasion attacks to construct adversarial examples for these models. Another evasion strategy is to mimic benign traffic behavior. We claim that mimicry traffic are still constrained by specific functionality (*e.g.* command-and-control traffic), which induces subtle deviations from benign features. Our approach can potentially capture these deviations due to the Transformer's ability to comprehensively model traffic features. If the original model fails to capture such features, we can further apply sequence masking or contrastive learning under perturbations, encouraging the model to learn a richer set of features that better cover potential mimicry behaviors. In the extreme case where malicious traffic perfectly aligns with the benign distribution, detection becomes inherently difficult. While rare, such attacks can be partially mitigated by incorporating host-level contextual signals (*e.g.* IP access sequences), which provide additional information beyond traffic features alone.

VII. RELATED WORK

Traditional Malicious Traffic Detection. Identifying the malicious network traffic has been extensively studied. Traditional arts [6], [37], [61], [63], [69], [85] mainly focus on extracting attack signatures or features from traffic payloads. For instance, [61] extracts five types of features from the URL and HTTP header and detects malware communication traffic through a template-matching approach. However, these traditional methods are unable to detect encrypted malicious traffic.

Encrypted Malicious Traffic Detection. Recently a series of malicious encrypted traffic detection methods [2], [3], [26], [48], [49], [50], [68], [71], [74], [80], [84], [92], [101], [104], [105] [11], [14], [52], [65], [94], [100] have been developed. Most of them share a similar design that profiles the behavior of encrypted traffic via manual feature engineering or automatic feature extraction and builds an ML/DL-based detector. For example, FS-Net [50] converts encrypted traffic into sequences of packet lengths and utilizes a novel

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

TABLE IX
THE F1 SCORE OF EACH METHOD ON THE REAL-WORLD DATASET

Training size	Noise Ratio	FS	Co+FS	DT+FS	ODDS+FS	SMOTE+FS	DT+ODDS+FS	ETA	DT+ODDS+ETA	RAPIER	RAPIER+
250	20%	.14 ± .20	.05 ± .16	.32 ± .28	.38 ± .19	.17 ± .06	.31 ± .25	.73 ± .28	.78 ± .30	.72 ± .04	**.79** ± .04
	25%	.30 ± .17	.08 ± .25	.26 ± .22	.23 ± .22	.16 ± .04	.22 ± .18	.49 ± .26	.65 ± .27	.70 ± .05	**.71** ± .06
	30%	.24 ± .13	.07 ± .20	.23 ± .15	.29 ± .15	.18 ± .04	.23 ± .13	.64 ± .34	.67 ± .30	.71 ± .06	**.73** ± .07
	35%	.23 ± .13	.25 ± .31	.24 ± .15	.25 ± .15	.18 ± .02	.21 ± .16	.47 ± .39	.49 ± .38	.67 ± .04	**.78** ± .10
	40%	.23 ± .12	.16 ± .23	.18 ± .13	.20 ± .11	.16 ± .03	.22 ± .13	.54 ± .33	.60 ± .35	.86 ± .04	.77 ± .07
	45%	.20 ± .09	.15 ± .11	.22 ± .09	.18 ± .09	.16 ± .02	.21 ± .09	.38 ± .32	.45 ± .35	.70 ± .08	**.75** ± .06
500	20%	.15 ± .18	.15 ± .15	.22 ± .14	.15 ± .16	.15 ± .05	.37 ± .25	.68 ± .39	.68 ± .39	.72 ± .05	**.75** ± .08
	25%	.22 ± .19	.20 ± .36	.15 ± .18	.11 ± .10	.13 ± .06	.16 ± .20	.51 ± .30	.58 ± .31	.76 ± .06	.75 ± .05
	30%	.24 ± .17	.11 ± .28	.24 ± .19	.23 ± .16	.13 ± .05	.25 ± .18	.48 ± .30	.58 ± .28	.75 ± .04	**.75** ± .10
	35%	.20 ± .14	.19 ± .38	.25 ± .18	.19 ± .20	.15 ± .03	.19 ± .16	.31 ± .34	.47 ± .32	.71 ± .02	**.81** ± .05
	40%	.18 ± .09	.17 ± .29	.19 ± .14	.28 ± .19	.11 ± .06	.18 ± .16	.36 ± .33	.45 ± .35	.76 ± .04	.76 ± .08
	45%	.20 ± .09	.12 ± .15	.17 ± .10	.17 ± .09	.13 ± .05	.17 ± .11	.16 ± .25	.17 ± .27	.77 ± .06	.76 ± .10
1000	20%	.30 ± .27	.00 ± .01	.29 ± .34	.27 ± .23	.13 ± .06	.24 ± .21	.73 ± .31	.66 ± .28	.81 ± .02	.73 ± .02
	25%	.36 ± .21	.00 ± .00	.29 ± .25	.23 ± .20	.15 ± .04	.21 ± .21	.55 ± .27	.44 ± .26	.64 ± .05	**.71** ± .05
	30%	.13 ± .15	.32 ± .46	.18 ± .15	.12 ± .16	.12 ± .05	.16 ± .14	.68 ± .34	.43 ± .33	.61 ± .01	**.74** ± .06
	35%	.20 ± .15	.41 ± .42	.28 ± .21	.25 ± .16	.12 ± .05	.30 ± .19	.50 ± .35	.45 ± .34	.69 ± .04	**.73** ± .06
	40%	.29 ± .14	.19 ± .30	.32 ± .21	.24 ± .18	.15 ± .01	.22 ± .18	.35 ± .33	.42 ± .37	.63 ± .02	**.73** ± .06
	45%	.14 ± .09	.04 ± .07	.22 ± .16	.17 ± .10	.14 ± .02	.17 ± .09	.25 ± .26	.35 ± .30	.68 ± .02	**.73** ± .07

deep-learning model to classify the traffic into different application categories. However, the effectiveness of these methods relies on a high-quality training set, where the training data is sufficient and correctly labeled. The detailed comparison of malicious traffic detection methods can be found in Appendix D.

VIII. CONCLUSION

We develop an encrypted malicious traffic system RAPIER+ that fully utilizes the different distributions of normal and malicious traffic data in the feature space to correct sample labels and augment new data for model training. RAPIER+ is a malware traffic detection system that simultaneously overcomes the challenges of training data insufficiency and label noises. We implement RAPIER+ and perform extensive evaluations based on two public datasets. The experimental results show average improvements of 358.5%, 314.0%, and 221.1% over the SOTA methods. We also experiment RAPIER+ with the dataset from a security enterprise, and it effectively achieves malicious traffic detection with the best F1 score of 0.81 and on average 288.7% improvements of the F1 score over the existing methods.

APPENDIX

A. The Detection Performance of All Methods

We evaluate the detection performance of all methods when the training sizes are 250 and 1000 and the label noise changes from 20% to 45%. We show the results of all methods on the DoHBrw, IDS, and IDS/DoHBrw datasets in Table VI, Table VII, and Table VIII, respectively. In general, our system achieves the best F1 score in almost all cases. In particular, when the training size is 250 and the noise ratio is 45%, the F1 score of our system is 0.59, 0.80, and 0.82 on the DoHBrw, IDS, and IDS/DoHBrw datasets, respectively. This shows an average improvement of 222.5%, 1330.0%, and 1523.6% over all baselines. The reason why our system outperforms other methods is that our system implements label noise correction and three different strategies of data augmentation for more accurate and robust malicious traffic detection.

B. The Comparison of Different Data Augmentation Methods

We compare the performance of different data augmentation methods when the training size is set to 250 and 1000. We utilize each method (except for NONE) to generate new training data, combine it with the original training data, and train the final detection model in our system to perform detection. The results on the three datasets are shown in Table X, Table XI and Table XII. It can be seen that our data augmentation module significantly outperforms other methods. The best F1 score of our module is about 0.95, 0.73, and 0.72 on the three datasets.

TABLE X
THE DETECTION PERFORMANCE WHEN USING DIFFERENT DATA AUGMENTATION METHODS ON THE DOHBRW DATASET

Training size	Metrics	NONE	ODDS	SMOTE	GAN	RAPIER	RAPIER+
250	P	.65 ± .17	.70 ± .13	.62 ± .13	.68 ± .18	.82 ± .05	**.84** ± .03
	R	.92 ± .06	.93 ± .06	.97 ± .01	.90 ± .08	.92 ± .05	**.93** ± .04
	F	.74 ± .12	.79 ± .08	.75 ± .09	.75 ± .11	.87 ± .02	**.88** ± .01
1000	P	.58 ± .16	.61 ± .18	.52 ± .16	.56 ± .17	.77 ± .05	**.78** ± .03
	R	.96 ± .05	.96 ± .05	.98 ± .01	.96 ± .03	.96 ± .02	**.96** ± .01
	F	.70 ± .13	.73 ± .14	.66 ± .16	.69 ± .14	.85 ± .03	**.86** ± .01

TABLE XI
THE DETECTION PERFORMANCE WHEN USING DIFFERENT DATA AUGMENTATION METHODS ON THE IDS DATASET

Training size	Metrics	NONE	ODDS	SMOTE	GAN	RAPIER	RAPIER+
250	P	.86 ± .15	.87 ± .25	.86 ± .12	.86 ± .15	.97 ± .02	**.98** ± .00
	R	.93 ± .07	.95 ± .01	.96 ± .00	.92 ± .06	.94 ± .01	**.98** ± .00
	F	.88 ± .10	.88 ± .21	.91 ± .06	.87 ± .12	.96 ± .01	**.98** ± .00
1000	P	.97 ± .03	.91 ± .22	.84 ± .27	.95 ± .08	.99 ± .00	.98 ± .00
	R	.95 ± .01	.96 ± .01	.97 ± .01	.95 ± .01	.97 ± .00	**.98** ± .00
	F	.96 ± .02	.91 ± .20	.86 ± .24	.95 ± .05	.98 ± .00	**.98** ± .00

TABLE XII
THE DETECTION PERFORMANCE WHEN USING DIFFERENT DATA AUGMENTATION METHODS ON THE IDS/DOHBRW DATASET

Training size	Metrics	NONE	ODDS	SMOTE	GAN	RAPIER	RAPIER+
250	P	.81 ± .27	.88 ± .22	.81 ± .31	.81 ± .25	.99 ± .01	**.99** ± .00
	R	.66 ± .35	.72 ± .37	.66 ± .38	.71 ± .34	.94 ± .03	**.97** ± .02
	F	.66 ± .34	.70 ± .36	.66 ± .36	.68 ± .31	.96 ± .01	**.98** ± .01
1000	P	.96 ± .05	.94 ± .13	.91 ± .18	.95 ± .06	.97 ± .03	**.98** ± .01
	R	.72 ± .28	.77 ± .26	.72 ± .28	.66 ± .32	.95 ± .05	**.97** ± .02
	F	.79 ± .21	.82 ± .22	.76 ± .26	.73 ± .28	.96 ± .03	**.98** ± .01

C. The Detection Performance on the Real-World Dataset

We conduct experiments on the real-world dataset with various training sizes (250, 500, and 1000) and noise ratio (20%, 25%, 30%, 35%, 40%, and 45%) settings, and the results are shown in Table IX. Overall, our system achieves the highest F1 score in almost all cases.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

TABLE XIII
THE COMPARISON OF MALICIOUS TRAFFIC DETECTION METHODS. STATS AND METADATA MEAN STATISTICS FEATURES AND TLS HANDSHAKE METADATA, RESPECTIVELY

Malicious traffic detection methods		Granularity	Input	Robustness under label noises	Limited training data
Traditional malicious traffic detection	[6], [61], [85]	HTTP Request	Payload	✗	✗
	[37]	HTTP Request	Payload	✗	✓
	[63]	Flow	Payload	✗	✗
	[69]	Packet	Payload	✗	✗
Encrypted malicious traffic detection	[2]	Flow	Stats & Metadata	✗	✗
	[3]	Flow	Stats & Packet Sequence & Metadata	✗	✗
	[26]	Flow Set	Metadata	✗	✗
	[52], [100]	Flow	Traffic Raw Byte	✗	✗
	[14], [48]	Flow	Traffic Raw Byte	✗	✓
	[49]	Flow	Packet Sequence & Metadata	✗	✗
	[50], [92], [94] [71], [11], [65]	Flow	Packet Sequence	✗	✗
	[80], [84]	Flow	Stats	✗	✗
	[101]	Flow Set	Packet Sequence	✗	✗
	[104], [105]	Flow	Packet Sequence	✗	✓
	RAPIER+	Flow	Packet Sequence	✓	✓

D. More Comparisons With Existing Methods

Malicious traffic detection has been extensively studied [2], [3], [6], [14], [26], [37], [48], [49], [50], [61], [63], [68], [69], [71], [74], [80], [84], [85], [92], [101], [104], [105], [11], [52], [65], [94], [100]. A number of detection methods have been developed. They detect malicious traffic by using different information. Unfortunately, they are unable to detect malicious traffic under limited training data with label noise. The detailed comparisons can be found in Table XIII.

REFERENCES

[1] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106944.

[2] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. ACM Workshop Artif. Intell. Secur.*, Oct. 2016, pp. 35–46.

[3] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1723–1732.

[4] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *Proc. ICML*, 2019, pp. 312–321.

[5] A. Bahramali, A. Bozorgi, and A. Houmansadr, "Realistic website fingerprinting by augmenting network traces," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2023, pp. 1035–1049.

[6] K. Bartos, M. Sofka, and V. Franc, "Optimized invariant representation of network traffic for detecting unseen malware variants," in *Proc. USENIX Secur.*, 2016, pp. 807–822.

[7] D. Bau et al., "Seeing what a GAN cannot generate," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4501–4510.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[9] L. Chen, Y. Wang, Q. Wang, Y. Song, and J. Chen, "Cybersecurity multi-dimensional few-shot data generation on malicious enhancement," *IEEE Trans. Depend. Secure Comput.*, vol. 22, no. 5, pp. 4988–4997, Sep. 2025.

[10] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1431–1439.

[11] Z. Chen, G. Cheng, D. Niu, Y. Zhao, Y. Zhou, and S. Jiang, "Ultimate encrypted traffic feature engineering: HTTPS encrypted traffic classification using restored application data unit length," *IEEE Trans. Depend. Secure Comput.*, vol. 23, no. 1, pp. 1290–1307, Jan. 2026.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[13] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1773–1788.

[14] T. Cui et al., "TrafficLLM: Enhancing large language models for network traffic analysis with generic traffic representation," 2025, *arXiv:2504.04222*.

[15] X. Deng, Q. Li, and K. Xu, "Robust and reliable early-stage website fingerprinting attacks via spatial–temporal distribution analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dec. 2024, pp. 1997–2011.

[16] X. Deng et al., "Robust multi-tab website fingerprinting attacks in the wild," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 1005–1022.

[17] X. Deng, Y. Zhang, Q. Li, Z. Liu, Y. Wang, and K. Xu, "Enhancing website fingerprinting attacks against traffic drift," in *Proc. NDSS*, 2026.

[18] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: The CICIDS2017 case study," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2021, pp. 7–12.

[19] A. Ferdowsi and W. Saad, "Generative adversarial networks for distributed intrusion detection in the Internet of Things," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[20] B. Frenay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, May 2014.

[21] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using GAN for improved liver lesion classification," in *Proc. IEEE 15th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2018, pp. 289–293.

[22] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," in *Proc. CCS*, 2021, pp. 3431–3446.

[23] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 452–467, Feb. 2023.

[24] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023, pp. 1–18.

[25] C. Fu, Q. Li, K. Xu, and J. Wu, "Point cloud analysis for ML-based malicious traffic detection: Reducing majorities of false positive alarms," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2023, pp. 1005–1019.

[26] Z. Fu et al., "Encrypted malware traffic detection via graph-based network analysis," in *Proc. 25th Int. Symp. Res. Attacks, Intrusions Defenses*, Oct. 2022, pp. 495–509.

[27] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked autoencoder for distribution estimation," in *Proc. PMLR*, 2015, pp. 881–889.

[28] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proc. AAAI*, 2017, pp. 1919–1925.

[29] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proc. ICLR*, 2017, pp. 1866–1874.

[30] I. Goodfellow et al., "Generative adversarial nets," in *Proc. NeurIPS*, 2014, pp. 2672–2680.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

[31] J. L. Guerra, C. Catania, and E. Veas, "Datasets are not enough: Challenges in labeling network traffic," *Comput. Secur.*, vol. 120, Sep. 2022, Art. no. 102810.

[32] B. Han et al., "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. NeurIPS*, 2018, pp. 8536–8546.

[33] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *Proc. ICIC*, 2005, pp. 878–887.

[34] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 1322–1328.

[35] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *Proc. NeurIPS*, 2018, pp. 10477–10486.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[37] S. T. K. Jan et al., "Throwing darts in the dark? Detecting bots with limited data using neural data augmentation," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1190–1206.

[38] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. icARTi*, 2000, pp. 111–117.

[39] J. Jia and N. Z. Gong, "AttriGuard: A practical defense against attribute inference attacks via adversarial machine learning," in *Proc. USENIX Secur.*, 2018, pp. 513–529.

[40] N. Karim, M. N. Rizve, N. Rahnavard, A. Mian, and M. Shah, "UNICON: Combating label noise through uniform selection and contrastive learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Washington, DC, USA: Combat, Jun. 2022, pp. 9666–9676.

[41] D. Kim, K. Ryoo, H. Cho, and S. Kim, "SplitNet: Learnable clean-noisy label splitting for learning with noisy labels," *Int. J. Comput. Vis.*, vol. 133, no. 2, pp. 549–566, Feb. 2025.

[42] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. NeurIPS*, 2014, pp. 3581–3589.

[43] A. H. Lashkari, G. Draper Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, 2017, pp. 253–262.

[44] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.

[45] J. Liang, W. Guo, T. Luo, V. Honavar, G. Wang, and X. Xing, "FARE: Enabling fine-grained attack categorization under low-quality labeled data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[46] Z. Liao, S. Hu, Y. Xie, and Y. Xia, "Instance-dependent label distribution estimation for learning with label noise," *Int. J. Comput. Vis.*, vol. 133, no. 5, pp. 2568–2580, May 2025.

[47] S. K. Lim, Y. Loo, N.-T. Tran, N.-M. Cheung, G. Roig, and Y. Elovici, "DOPING: Generative data augmentation for unsupervised anomaly detection with GAN," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 1122–1127.

[48] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 633–642.

[49] C. Liu, Z. Cao, G. Xiong, G. Gou, S.-M. Yiu, and L. He, "MaMPF: Encrypted traffic classification based on multi-attribute Markov probability fingerprints," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Jun. 2018, pp. 1–10.

[50] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1171–1179.

[51] S. Liu, Z. Zhu, Q. Qu, and C. You, "Robust training under label noise by over-parameterization," in *Proc. ICML*, 2022, pp. 14153–14172.

[52] Y. Liu, X. Wang, B. Qu, and F. Zhao, "ATVITSC: A novel encrypted traffic classification method based on deep learning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 9374–9389, 2024.

[53] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Proc. NeurIPS*, 2014, pp. 855–863.

[54] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

[55] Y. Lyu and I. W. Tsang, "Curriculum loss: Robust learning and generalization against label corruption," in *Proc. ICLR*, 2019, pp. 1–22.

[56] E. Malach and S. Shalev-Shwartz, "Decoupling 'when to update' from' 'how to update,'" in *Proc. NeurIPS*, 2017, pp. 960–970.

[57] Q. Meng et al., "When unknown threat meets label noise: A self-correcting framework," *IEEE Trans. Depend. Secure Comput.*, vol. 23, no. 1, pp. 905–922, Jan. 2026.

[58] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, "Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 1009–1024.

[59] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[60] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari, "Detection of DoH tunnels using time-series classification of encrypted traffic," in *Proc. IEEE Int. Conf. Depend., Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2020, pp. 63–70.

[61] T. Nelms, R. Perdisci, and M. Ahamad, "ExecScent: Mining for new C&C domains in live networks with adaptive control protocol templates," in *Proc. USENIX Secur.*, 2013, pp. 589–604.

[62] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2233–2241.

[63] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, Dec. 1999.

[64] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *Proc. USENIX Secur.*, 2019, pp. 729–746.

[65] J. Piet, A. Sharma, V. Paxson, and D. Wagner, "Network detection of interactive $*SSH*$ impostors using deep learning," in *Proc. USENIX Secur.*, 2023, pp. 4283–4300.

[66] Y. Qian et al., "Enhancing resilience in website fingerprinting: Novel adversary strategies for noisy traffic environments," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 7216–7231, 2024.

[67] Y. Qing et al., "Low-quality training data only? A robust framework for detecting encrypted malicious network traffic," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2024, pp. 1–18.

[68] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[69] M. Roesch, "Snort-lightweight intrusion detection for networks," in *Proc. LISA*, 1999, pp. 229–238.

[70] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.

[71] M. Shen, Z. Gao, L. Zhu, and K. Xu, "Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Jun. 2021, pp. 1–10.

[72] M. Shen, K. Ji, Z. Gao, Q. Li, L. Zhu, and K. Xu, "Subverting website fingerprinting defenses with robust traffic representation," in *Proc. USENIX Secur.*, 2023, pp. 607–624.

[73] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[74] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1928–1943.

[75] H. Song, M. Kim, and J.-G. Lee, "SELFIE: Refurbishing unclean samples for robust deep learning," in *Proc. ICML*, 2019, pp. 5907–5915.

[76] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8135–8153, Nov. 2023.

[77] S. Y. Ali and H. Maseeh, "Dropout: An effective approach to prevent neural networks from overfitting," *Asian J. Res. Comput. Sci.*, vol. 18, no. 2, pp. 163–185, Feb. 2025.

[78] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *Proc. ICLR*, 2015, pp. 1–11.

[79] R. Tang et al., "ZeroWall: Detecting zero-day web attacks through encoder–decoder recurrent neural networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 2479–2488.

[80] F. Tegeler, X. Fu, G. Vigna, and C. Kruegel, "BotFinder: Finding bots in network traffic without deep packet inspection," in *Proc. 8th Int. Conf. Emerg. Netw. Experiments Technol.*, Dec. 2012, pp. 349–360.

[81] M. L. Hazelton, "Variable kernel density estimation," *Austral. New Zealand J. Statist.*, vol. 45, no. 3, pp. 271–284, Sep. 2003.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.

[82] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 5998–6008.
[83] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu, "Suppressing model overfitting in mining concept-drifting data streams," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2006, pp. 736–741.
[84] S. Wang et al., "TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic," in *Proc. IEEE/ACM 24th Int. Symp. Quality Service (IWQoS)*, Jun. 2016, pp. 1–6.
[85] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Detecting Android malware leveraging text semantics of network flows," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1096–1109, May 2018.
[86] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surveys*, vol. 53, no. 3, pp. 1–34, May 2021.
[87] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 322–330.
[88] WatchGuard Technologies Threat Lab. (2023). *Internet Security Report Q1 2023*. Watchguard Threat Lab Internet Security Report. Internet Security Report Q1 2023 by WatchGuard Technologies. [Online]. Available: https://www.watchguard.com/wgrd-resource-center/security-report-q1-2023
[89] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using CWSandbox," *IEEE Secur. Privacy*, vol. 5, no. 2, pp. 32–39, Apr. 2007.
[90] X. Wu, W. Guo, J. Yan, B. Coskun, and X. Xing, "From grim reality to practical solution: Malware classification in real-world noise," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 2602–2619.
[91] X. Xia et al., "Part-dependent label noise: Towards instance-dependent label noise," in *Proc. NeurIPS*, 2020, pp. 7597–7610.
[92] R. Xie et al., "Rosetta: Enabling robust TLS encrypted traffic classification in diverse network environments with TCP-aware traffic augmentation," in *Proc. ACM Turing Award Celebration Conf. - China*, Jul. 2023, pp. 131–132.
[93] J. Xu, Y. Li, and R. H. Deng, "Differential training: A generic framework to reduce label noises for Android malware detection," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–14.
[94] S.-J. Xu, G.-G. Geng, X.-B. Jin, D.-J. Liu, and J. Weng, "Seeing traffic paths: Encrypted traffic classification with path signature features," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2166–2181, 2022.
[95] Y. Yao et al., "Dual T: Reducing estimation error for transition matrix in label-noise learning," in *Proc. NeurIPS*, 2020, pp. 7260–7271.
[96] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption?," in *Proc. ICML*, 2019, pp. 7164–7173.
[97] Q. Yuan, G. Gou, Y. Zhu, Y. Zhu, G. Xiong, and Y. Wang, "MCRe: A unified framework for handling malicious traffic with noise labels based on multidimensional constraint representation," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 133–147, 2024.
[98] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient GAN-based anomaly detection," 2018, *arXiv:1802.06222*.
[99] M. Zhan et al., "FlowRefiner: A robust traffic classification framework against label noise," in *Proc. NeurIPS*, 2025, pp. 1–12.
[100] H. Zhang et al., "TFE-GNN: A temporal fusion encoder using graph neural networks for fine-grained encrypted traffic classification," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 2066–2075.
[101] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "HoMonit: Monitoring smart home apps from encrypted traffic," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1074–1088.
[102] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. NeurIPS*, 2018, pp. 8792–8802.
[103] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial networks," in *Proc. ICLR*, 2017, pp. 1086–1102.
[104] R. Zhao, X. Deng, Z. Yan, J. Ma, Z. Xue, and Y. Wang, "MT-FlowFormer: A semi-supervised flow transformer for encrypted traffic classification," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 2576–2584.
[105] W. Zheng, C. Gou, L. Yan, and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *Proc. Web Conf.*, Apr. 2020, pp. 13–22.
[106] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *Proc. USENIX Secur.*, 2023, pp. 6203–6220.

Yuqi Qing (Student Member, IEEE) is currently pursuing the Ph.D. degree with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China. His research interests include network security, with a particular focus on robust malicious traffic detection.

Qilei Yin received the Ph.D. degree in information security from the University of Chinese Academy of Sciences in 2020. He is currently an Associate Research Professor at the Zhongguancun Laboratory. Previously, he was with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include network security, with a particular focus on malicious traffic detection.

Xinhao Deng received the Ph.D. degree from the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China, in 2025. He is currently a joint Post-Doctoral Researcher with Tsinghua University and the Ant Group, China. His research interests include AI security and network security, with a particular focus on agent security and website fingerprinting.

Yihao Chen is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include the Internet routing security, BGP analysis, and network measurement. He was a recipient of the USENIX Security 2024 Distinguished Paper Award.

Zhuotao Liu (Senior Member, IEEE) received the Ph.D. degree from the University of Illinois at Urbana–Champaign (UIUC), IL, USA. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China. His research interests include network security and trusted AI systems.

Kun Sun (Senior Member, IEEE) received the Ph.D. degree from North Carolina State University, NC, USA. He is currently a Professor with the Department of Information Sciences and Technology, George Mason University (GMU), VA, USA, where he is also the Director of the Center for Information Security and Assurance. His research interests include systems and network security.

Ke Xu (Fellow, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China, in 2001. He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University. He has published over 100 articles in leading venues in networking and security. He is a Fellow of China Institute of Electronics.

Jia Zhang received the Ph.D. degree from Tsinghua University, Beijing, China, in 2010. He is currently an Associate Research Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include network protocol vulnerability and Internet infrastructure security.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China, in 2012. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include Internet security, the IoT security, and machine learning security. He currently serves on the editorial board for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, ACM TOPS, and ACM DTRAP.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on November 15,2026 at 03:08:13 UTC from IEEE Xplore. Restrictions apply.