Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1556768, 14 pages
https://doi.org/10.1155/2022/1556768

Research Article

AS-DMF: A Lightweight Malware Encrypted Traffic Detection Method Based on Active Learning and Feature Selection

Yuehua Huo,[1,2] **Faqi Zhao**,[1] **Hangsheng Zhang**,[3,4] **Shangyuan Zhuang**,[3,4] **and Jiyan Sun**[3]

[1]*School of Mechanical Electronic & Information Engineering, China University of Mining and Technology-Beijing, Beijing 100083, China*
[2]*Network and Information Center, China University of Mining and Technology-Beijing, Beijing 100083, China*
[3]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[4]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China*

Correspondence should be addressed to Yuehua Huo; huoyh@cumtb.edu.cn

Received 31 May 2022; Accepted 6 August 2022; Published 21 September 2022

Academic Editor: Suhua Tang

The sharp increasing volume of encrypted traffic generated by malware brings a huge challenge to traditional payload-based malicious traffic detection methods. Solutions that based on machine learning and deep learning are becoming mainstream. However, the machine learning-based methods are limited by manual-design features, which have the problem of highly correlated multicollinearity. And both methods rely heavily on a large number of labeled samples, which needs lots of human effort. In this paper, we apply the active learning to the malicious encrypted traffic detection problem and propose AS-DMF framework. AS-DMF is a lightweight detection framework that combine the uncertainty sampling and density-based query strategy to query the informative and representative instances from the sample set and then train them in a detection (DMF) model. Moreover, we propose a feature selection mechanism which can select the meaningful features of traffic efficiently. Our comprehensive experiments on the real-word dataset indicate that AS-DMF achieves lightweighting at both feature and data levels with a high performance of 0.9460 mAcc.

1. Introduction

With the increasing awareness of user privacy protection, many encryption protocols are widely used by applications, especially TLS (Transport Layer Security) [1]. Unfortunately, encryption protocols also give malware the opportunity to hide malicious behavior. This is because the contents of communication are not available, making malicious encrypted traffic hard to detect by network intrusion detection system. As a result, malware traffic using TLS encryption has gradually increased. According to the report of Sophos News, in 2020, 23% of malware communicating with remote systems over the Internet used TLS encryption, while this number is closer to 46% today [2]. Although many solutions to detect malicious encrypted traffic have been proposed, they are still difficult to deploy in real-word environments [3]. On the one hand, solutions that decrypt TLS traffic weaken user privacy and are limited to use [4]. On the other hand, the complexity of the detection system increases the difficulty of deployment [5]. Therefore, lightweight TLS malicious encrypted traffic detection becomes a focus issue and attracts the widespread attention of industries and academia.

Combining machine learning algorithms and statistical features extracted from traffic flows becomes the mainstream method for malicious encrypted traffic detection [6]. The general procedures include feature engineering and model training [7]. Researchers are able to obtain not only statistical characteristics from the traffic level, but plaintext information from TLS handshake packets as features to train the model. However, the manually designed features have the problems that their own changes have little influence on the target and multicollinearity among them. And the detection models are also needed to be carefully selected to adapt the features. Obviously, the expressiveness of this kind method depends on model and feature selection,

and the result of each factor will directly affect the final detection performance. Due to the drawbacks of machine learning based solutions above, deep learning-based solutions are designed [8]. Researchers have mainly used the autoencoder, CNN, and LSTM networks to learn patterns from traffic data by extracting observable information from the TLS handshake [9, 10]. However, the features automatically extracted by deep learning network are poorly interpretable, and those methods rely excessively on a large amount of labeled data.

A common limitation of machine learning and deep learning applications is the need for a large number of accurately labeled samples to provide the model for training. However, it is expensive to obtain labeled encrypted traffic in real networks. An alternative idea to solve this problem is to introduce a pool-based active learning approach. The objective of active learning is to train an accurate prediction model with minimum cost by labeling the most informative and representative instances. Initially, the training set with labels is small. The learner consults the oracle to obtain the class labels of the query instances, which are added to the training set to update the model. This process is repeated until the model achieves the preset indicator values. The fundamental question is how active learners pick the most critical instances to label. The uncertainty sampling query strategy often builds a model using available labels with the least certainty to predict others [11]. However, some instances at the classification boundary may be outliers and not suitable as training samples, although with high uncertainty. Moreover, the strategy does not consider the distribution of the original data when querying the samples. We believe that informative and representative instances are not only samples with strong uncertainty, but instances with dense distribution (high information density) in the sample domain space. In that case, samples with higher density in the spatial region should be taken into account when selecting instances.

In this paper, we propose an AS-DMF detection framework based on active learning for malicious encrypted traffic detection. The major structure of AS-DMF framework include a feature selection mechanism to select the important features, a query strategy combining uncertainty sampling and density-based approach to query the most informative and representative instances for oracle to label, and a detection model called double layer mutimodel fusion (DMF) classifier to train the labeled instances. The features for detection are selected from initial feature set by feature selection mechanism and are trained by the DMF classifier. The principle of this mechanism is to use analysis of variance (ANOVA) method and mutual information (MI) to select features that are informative and less relevant. The detection DMF classifier is build based on stacking strategy to reconstruct and train the selected features. The contributions of this paper are summarized as follows.

(1) We propose a lightweight AS-DMF framework for malicious encrypted traffic detection. AS-DMF framework jointly uses a few number of instances with a small feature set to identify flows and consists of a feature selection mechanism, a DMF classifier, and a query strategy

(2) The feature selection mechanism is used to select the important features. By combining ANOVA and MI, the selected features can become more discriminative, which improves the detection performance

(3) Our DMF classifier with feature selection mechanism achieves excellent results on real-word network traffic for malicious encrypted traffic detection, and AS-DMF framework achieves lightweighting at both the data and feature level. The source code can be found at the link: https://github.com/Timeless-zfqi/AS-DMF-framework

The remainder of this article is organized as follows.

Section 2 reviews the related work on TLS malicious traffic detection method and active learning query strategy. Section 3 describes the proposed DMF classifier and AS-DMF framework. In Section 4, we conduct experiments on real-world datasets to evaluate the performance of our method and compare with advanced methods. Section 5 concludes this article and discusses future directions.

2. Related Work

2.1. Feature Engineering and Malicious Encrypted Traffic Classification. The TLS protocol can provide a series of observable data during the handshake process, as shown in Figure 1. Therefore, many machine learning approaches based on the features of handshake packets have been proposed to predict TLS encrypted traffic, aiming to distinguish benign encrypted traffic from malicious encrypted traffic [12]. Cisco proposed a "data omnia" machine learning approach to correlated TLS handshake, DNS, and HTTP header information to achieve 99.99% accuracy using L1 logistic regression [13]. Pai et al. implemented a decision tree algorithm for traffic detection using the *Client Hello* feature in the TLS handshake record [14]. Anderson and McGrew analyzed two causes of inaccurate ground truth and a highly nonstationary data distribution that affect the application of machine learning models [3]. Their works show us the decisive role of feature engineering for machine learning. Therefore, many classification methods based on multifeature fusion are explored. Yang et al. proposed a stacking-based ensemble learning method MGEL for multi-granularity features to identify encrypted malicious traffic [15]. The X.509 certificates are complex and diverse [16]. Torroledo et al. proposed a neural network model in combination with LSTM networks to verify that certificates are significantly differentiated [10]. Yu et al. proposed a method that combines natural language processing and machine learning to learn the laws of data using TF-IDF [17]. However, these methods are only considering feature diversity, but the problem of feature correlation is not effectively addressed.

Deep learning methods have shown their effectiveness in many applications including computer vision, human activity recognition, and network traffic classification [18]. Yu

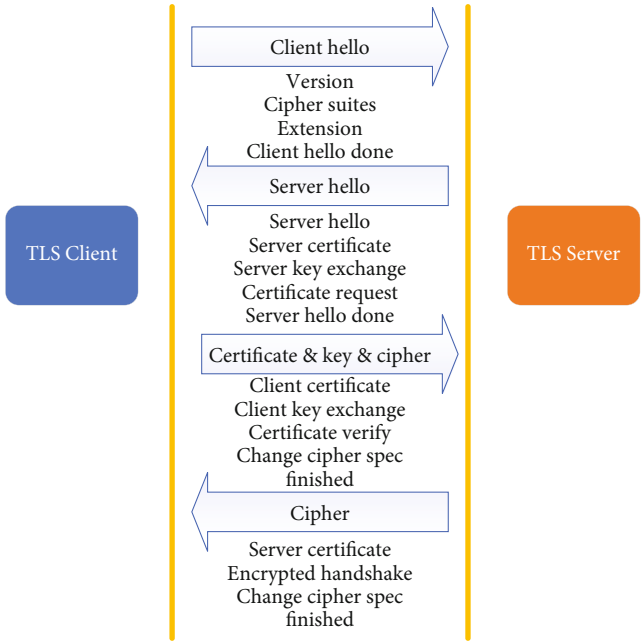

Figure 1: TLS handshake process. Graphical representation of a simplified TLS handshake and the features used in this paper are taken from the *Client Hello*, *Server Hello*, *Certificate*, and *Cipher* messages.

et al. used an autoencoder network to reconstruct features from TLS traffic [17]. Zhang et al. converted traffic into 2D image and combined CNN and LSTM network to create a new neural network [19]. The model achieves an average accuracy of 99.5% on TLS encrypted traffic classification. Chen et al. proposed a tree-shaped deep neural network (TSDNN) to detect malicious traffic when faced with the problem of gradient dilution due to unbalanced data [20]. However, they all use an end-to-end model to classify behaviors, but do not consider the complexity of the practical application.

2.2. Active Learning. The goal of active learning is to reduce the cost of labeled data [21]. The active learner can query and train a small number of samples to enable the model to achieve high performance [22]. In order to obtain a lightweight training set, the active learner needs to sample the initial sample set. Sampling strategies can be divided into exploitative (uncertainty-based), explorative (density-based), or a combination of both [23]. And most commonly used to query the least uncertain data points is the strategy based on exploitative. Despite the success of these methods, the problem of not being able to take the sample distribution into account also exists.

In contrast, the underlying data distribution of the unlabeled data can be considered by exploration-driven strategy. To reduce the query number and obtain better performance, it is necessary to dynamically select instances by considering representativeness and informativeness. Zha et al. investigated active learning-based anomaly detection and proposed a novel framework called Active Anomaly Detection with MetaPolicy (Meta-AAD) [24]. This study provides strong support for us to introduce active learning into malicious encrypted traffic detection. Wang et al. proposed an active learning method based on density clustering (ALEC) that takes the structure of the data into account and selects the most representative instances [25]. And then, Shi et al. proposed an active density peak (ADP) clustering algorithm that considers both representativeness and informativeness, while informative instances are queried to reduce the uncertainty of clustering results [26]. These algorithms are based on clustering to ensure the minimum distance between two samples. Konyushkova et al. suggested a novel data-driven approach (LAL) to active learning. This approach formulates the query selection procedure as a regression and works well on real data [27]. Huang et al. addressed the limitation of ad hoc in finding unlabeled instances by developing a principled approach, termed QUIRE, based on the min-max view of active learning [28]. This approach provides a systematic way to measure and combine the informativeness and representativeness of an unlabeled instance. However, these methods ignore the information of global uncertainty and sensitive to outliers.

In this paper, we combine uncertainty sampling and density-based strategies to query instances in the sample set. The new query strategy uses the Euclidean distance as a density calculation metric. The benefit of this design is to reduce the impact generated by outliers while obtaining the most representative and informative instances.

3. Model Framework and Methods

In this section, we first give the categories and contents of the defined features. Then, the feature selection mechanism, DMF classifier, and AS-DMF framework are introduced briefly.

3.1. Feature Selection and Numericalization. To distinguish malicious traffic from benign traffic encrypted by TLS, we extract the following 3 class features from captured packets. The open source tool Zeek is used to complete feature extraction. These features are stored in *flowmeter.log*, *conn.log*, *ssl.log*, and *X509.log* in log format. The unique identifiers (uid) for each flow are assigned by Zeek, which are used to track the relevant activity of the flow in each log.

(1) *Flow metadata and connection features.* The first class of features is the connection status and flow metadata stored in *flowmeter.log* and *conn.log*. The connection state is associated with the TCP (Transmission Control Protocol) stateful. And the flow metadata include number of packets, the proportion of backward and forward packets number, the number of bytes of the headers, payload size, the total number of FIN/SYN/PSH/ACK/URG/CWR/ECE flags which have been seen in a TCP flow, the interarrival time between two consecutive packets, the number of payload bytes transmitted in a bulk transmission, and the duration of the flow. The first class of features reflects the statistical behavior of the

traffic and contains a total of 89 dimensions of numerical type data

(2) *Unencrypted TLS handshake features*. The second class of features is the TLS handshake logs stored in *ssl.log*. These features include the TLS protocol version, cipher suite, TLS extension, and connection status extracted from the *Client Hello* and *Server Hello*. In addition, the length of the public key of the cipher group extracted from the *client key exchange massage* is included. A bag-of-words model is used to encode the cipher suite and transform the words into a frequency-of-words matrix. And the rest of features are mapped to a number field and normalized to a uniform scale

(3) *X.509 certificate features*. The third class of features is the TLS certificate information stored in *X509.log*, including the certificate subject, issuer, and certificate chain. The traditional one-hot encoding approach for certificate features leads to dimensional catastrophes. The certificate features that we extract have a common paradigm. For example, the *subject* field is fixed as $\{\mathrm{CN} = *, \mathrm{OU} = *, \mathrm{O} = *, \mathrm{L} = *, \mathrm{ST} = *, \mathrm{C} = *\}$. Therefore, we encode the certificate features using the TF-IDF method. In the paradigm, the words "CN," "OU," "O," "L," "ST," and "C" are inherent and do not change with the sample. In our experiments, we use these words to combine features in the encoding phase and remove them after encoding

3.2. Feature Selection Mechanism. For machine learning, the number and quality of features directly affect the efficiency of model learning. Therefore, it is essential to reduce the feature dimension. The main weak point of the initial feature set is the high correlation and multicollinearity problem, which cannot improve the detection model performance. We select the extracted features based on 3 main aspects: (1) the volatility of the features themselves, (2) the relevance of the features to the target, and (3) the correlation between the features. ANOVA is an efficient feature selection method, which drop the useless features. And the MI method traces the informative features with maximal information coefficient (MIC).

According to the above principle, the feature selection mechanism can be described as the following 3 steps. First, observe the distribution of individual features from the sample set. Features with highly concentrated, that is, small volatility in spatial distribution, should be removed, due to it is difficult to classify samples according to these features.

Second, the variance of each feature is calculated by ANOVA, as shown in

$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}, \tag{1}$$

where σ^2 represents the variance, μ is the mean of a feature, and the number of samples is N. A variance threshold is set to measure the correlation of the features with the target by analyzing the effect of perturbations from different sources on the results. And a feature with a low variance value below the threshold should be dropped from the feature set.

Third, features with low correlation within the feature set are selected. The MI can map high-dimensional features to the optimal vector comparison space, and the projected samples have maximum differentiability in the subspace. The MI is defined as follows:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} = \int p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} dxdy, \tag{2}$$

where $I(X;Y)$ represents the correlation between X and Y. Moreover, the MIC is able to handle both linear and nonlinear features and treating each feature equally.

$$\mathrm{MIC}(x;y) = \max_{x \cdot y < B} \frac{I(x;y)}{\log_2 \min(x,y)}. \tag{3}$$

Therefore, the features with top MIC values should be selected to form a lightly feature set.

3.3. DMF Model Structure. To identify malicious encrypted traffic in the network, as shown in Figure 2, we design a DMF classifier. Machine learning models differ in their ability to learn from linear and nonlinear data. To achieve a better detection performance, DMF combines three heterogeneous models. Specifically, random forest classifier, XGBoost classifier, and Gaussian Naive Bayes classifier are used to learn the patterns in the dataset. The three classifiers are combined with a logistic regression using the stacking strategy. The working mechanism of the DMF classifier is as follows.

First, the three classifiers, as the primary learners of the DMF classifier, are used to learn and reconstruct the features in Section 3.1. And we train primary learners using a 5-fold cross-validation approach. The predictions of the 5-fold cross-validations performed on training set D are reconstructed into new 3-dimensional features, while the corresponding labels of the training samples remain unchanged. And then, a new training set D' (level 1) is formed by the new features with the original labels. Finally, a logistic regression model (level 2), as a secondary learner, is used to train D' so that the DMF classifier outputs the predicted labels of the samples. It is worth to notice that we do not average the probability values generated by each category, as this would be unfair to the primary learner with high reliability. We use the probability values of the categories as input to the secondary learner, and we use logistic regression to train the training set D' to dynamically adjust the weights of the primary learners.

3.4. Active Learning-Based AS-DMF Framework. AS-DMF framework compresses the raw encrypted traffic and learn instances that can represent the main points of the raw encrypted traffic under the query strategy. In order to achieve this recurrence, the query strategy of AS-DMF needs to search the most important instances as the informativeness and representativeness samples of the raw encrypted traffic.

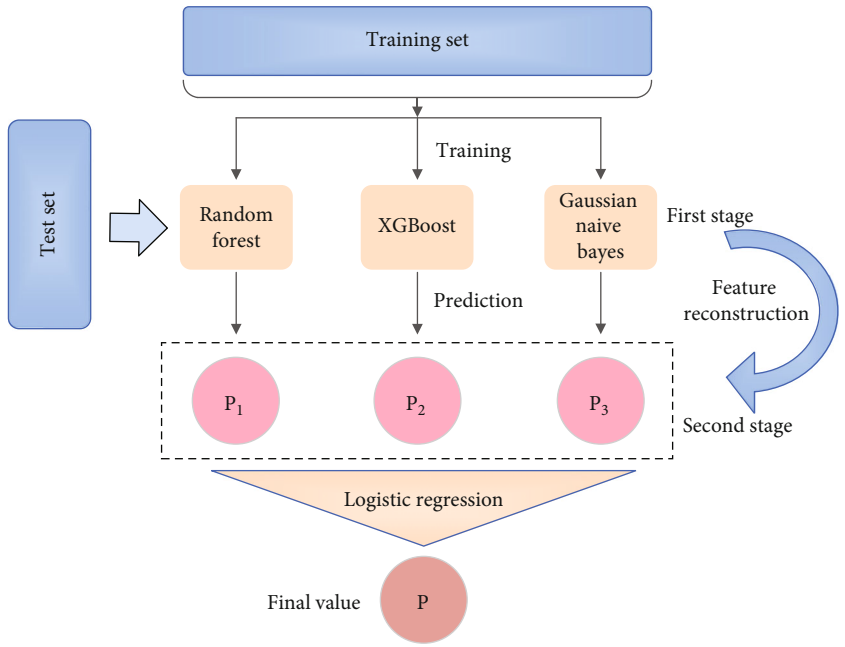

Figure 2: DMF classifier structure.

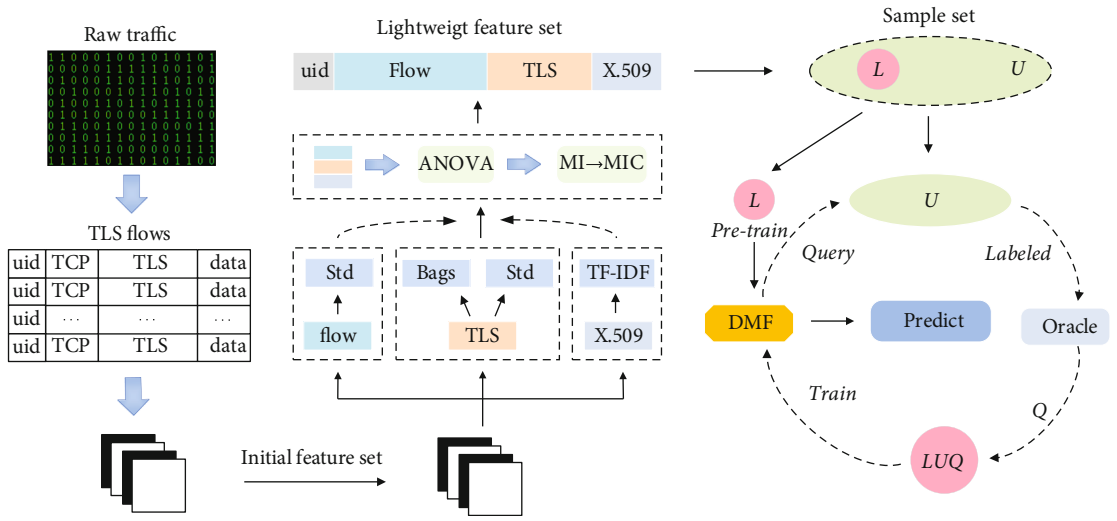

Figure 3: AS-DMF framework. The "Std" represents standardization, and "Bags" represents bag-of-words method.

(1) *AS-DMF framework.* The architecture of AS-DMF framework is shown in Figure 3. Feature selection mechanism, DMF classifier, pool-based active learning approach, and query strategy are four elements of AS-DMF framework. First, the TLS flows are filtered from raw traffic for initial feature extraction. Then, the feature selection mechanism selects the main features to construct a lightweight feature set. The data that retain these features form a sample set, and a part of samples are labeled to obtain a small training set $\mathscr{L}$ for pretraining the DMF classifier. Finally, the pool-based active learning approach uses a query strategy to search the most informative and representative instances of $\mathscr{U}$, which is a large number of unlabeled sample sets, to label for DMF classifier training. The pool-based active learning query process is shown in Algorithm 1

(2) *Problem.* The UNSM query strategy based on exploitative is commonly used because of its simplicity and speed. In ideal, as shown in Figure 4(c-1), an active learner uses a metric function to derive the uncertainty value of each instance and determines whether to discard the instance by setting a threshold. These metric values form an uncertainty region, and active learner

Input: Labeled set $\mathcal{L}$, unlabeled pool $\mathcal{U}$,
query strategy $\phi(\cdot)$, query batch size B, learner M
do
// *learn a M using the current* $\mathcal{L}$
$M = M.\text{train}(\mathcal{L})$;
// *initial the empty query batch*
$\mathcal{Q} = [\,]$;
for $b = 1$ to B **do**
// *select the most informative and representative instance*
$x_b^* = \arg\max_{x \in u} \phi(x)$;
// *move the selected query instance from* $\mathcal{U}$ *to* $\mathcal{Q}$
$\mathcal{Q} = \mathcal{Q} \leftarrow x_b^*$;
$\mathcal{U} = \mathcal{U} - x_b^*$;
end for
// *label each instance in* $\mathcal{Q}$ *and add to* $\mathcal{L}$
for $i = 1$ to len($\mathcal{Q}$) **do**
$y = \text{label}(\mathcal{Q}[i])$;
$\mathcal{L} = \mathcal{L} \leftarrow (x_i, y)$;
end for
until *reached the preset number of queries*;

Algorithm 1: Pool-based active learning query process.

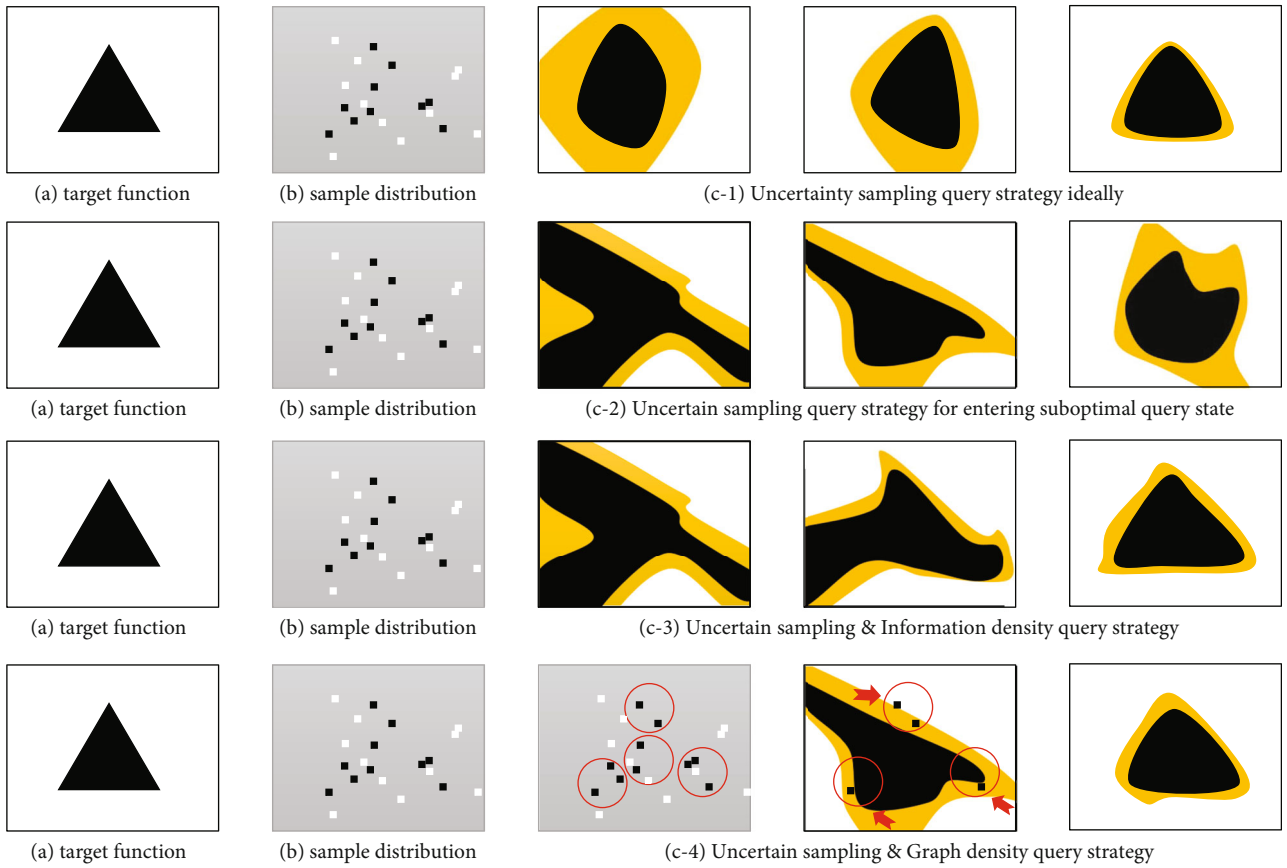

Figure 4: Fitting results of the three query strategies on the same data distribution with the query on the objective function (black area represents positive cases, white area represents negative cases, and yellow area represents uncertainty region).

Table 1: The statistical information of dataset.

	Type	Flow
Malware	Dridex	4969
	Tickbot	5045
	Trojan-Rasftuby	55
	Dyname	5154
	Bunitu	2708
	Cobalt	250
	Yakes	1819
Benign	Benign	20000

Table 2: Performance of DMF classifier on different feature sets.

	Accuracy	Precision	Recall	F_1 measure	FPR
F_{10}	0.9793	0.9691	0.9901	0.9795	0.0315
F_{15}	0.9854	0.9762	0.9950	0.9855	0.0241
F_{20}	0.9858	0.9764	0.9956	0.9859	0.0210
F_{25}	0.9864	0.9765	0.9968	0.9865	0.0240
F_{30}	0.9872	0.9774	0.9975	0.9873	0.0230
F_{35}	0.9882	0.9792	0.9976	0.9883	0.0211
F_{40}	0.9895	0.9832	0.9961	0.9896	0.0170
F_{45}	0.9896	0.9833	0.9962	0.9897	0.0168
F_{50}	0.9916	0.9858	0.9976	0.9917	0.0143
F_{all}	**0.9949**	**0.9917**	**0.9981**	**0.9949**	**0.0080**

select instances within the region to be labeled by oracle. However, this strategy assumes only a single hypothesis space and lacks consideration of the data structure. For the former, the metric function only considers the hypothesis space of the previous state, which tends to make the query instance deviate from the sample center. This problem, as shown in Figure 4(c-2), will lead our model to stay away from the objective function. For the latter, the UNSM query strategy takes the data structure out of account, which is prone to querying outliers simply. However, the density-based approach considers the data distribution of the unlabeled samples in the pool

In order to address the above issues, we design a density-based query strategy. Considering that different scenes have different degrees of attention to the local and global samples density, we design the query strategy based on information density for scenarios that pay more attention to local density, and the query strategy based on graph density for scenarios that pay more attention to global density, respectively. The details will be presented in the following.

(3) *Uncertainty sampling and information density.* An instance as an outlier located on a classification boundary, it does not represent other instances in the distribution, although it has high uncertainty. Representative instances should not only be uncertain instances but also instances that are representative of the input distribution. Therefore, the input distribution is modeled using the information density-based strategy. For an unlabeled dataset X_u, as shown in Figure 4 (c-3), the information density of an instance x can be calculated as:

$$I(x) = \frac{1}{|X_u|} \sum_{x' \in X} \text{sim}\left(x, x'\right), \tag{4}$$

where X_u represents all instances in $\mathcal{U}$, x' represents instances in $\mathcal{U}$ except for x, and $\text{sim}(x, x')$ is a Euclidean similarity function, which is the reciprocal of Euclidean distance. That means the higher the information density, the more similar the given instance is to the rest of the data. Then, the uncertainty measures weighted by information density as

$$x_I^* = \arg\max_x \phi^E(x) \times I^\beta(x), \tag{5}$$

where x_I^* represents the instance with maximum information, $\phi^E(x)$ represents the entropy method for uncertain sampling, and β is the exponential parameter to control the relative importance of the density $I(x)$. The samples in the pool are weighted to lead the query proceeds in the direction of high density. Moreover, the uncertainty region is reduced to guide the query results to approximate the true data distribution.

(4) *Uncertainty sampling and graph density*. The graph density-based approach considers the global data distribution. In comparison to information density, the basic principle of graph density is to use the graph structure to identify connected nodes (function node):

$$\text{Node}(x_i) = \sum_{j=1}^{n} \exp\left(-\frac{d^2(x_i, x_j)}{4{r_a}^2}\right), \tag{6}$$

where $d(x_i, x_j)$ denotes the Euclidean distance between two samples and r_a is the neighborhood of node. First, it calculates the minimum distance $D_t(x_i)$ of an unlabeled sample in $\mathcal{U}$ to all labeled sample in $\mathcal{L}$.

$$D_t(x_i) = \min_{x_j \in L} d\left(x_i, x_j\right). \tag{7}$$

Then, select the farthest instance $x^* = \arg\max_i (D_t(x_i))$ for labeling.

First, we build a graph of the k-nearest neighbor containing k samples and compute the Euclidean distance d between the instance x_i in $\mathcal{U}$ and the k samples in the graph. A symmetric connection matrix P will be created if d is one of the k smallest Euclidean distances (or x_i is the k nearest

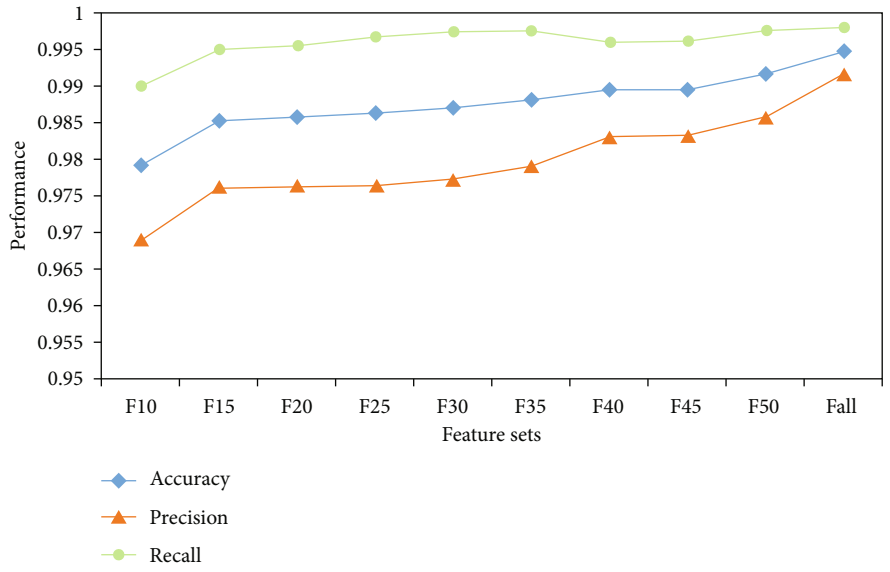

Figure 5: The performance trends of DMF in accuracy, recall rate, and precision.

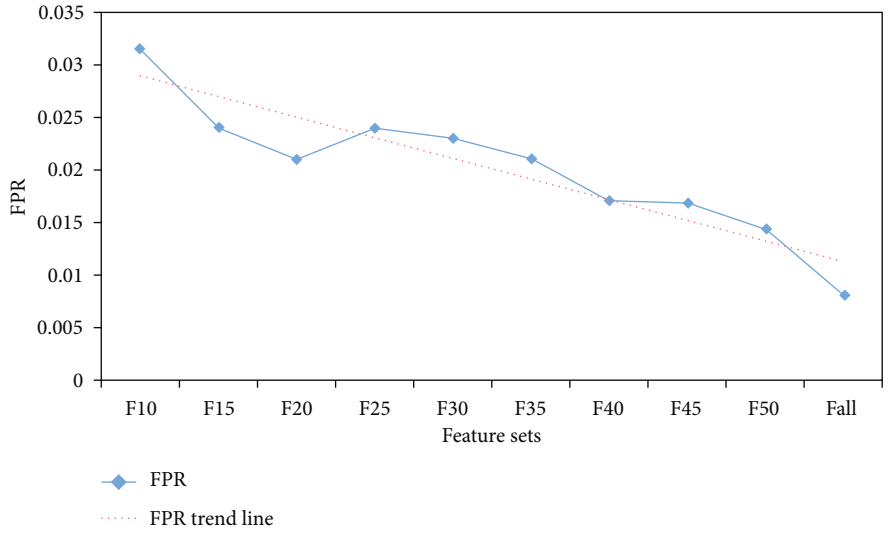

Figure 6: Trend of FPR under different feature sets.

neighbor of one of the labeled samples), initializing the matrix $P_{ij} = 1$. The connectivity matrix is updated after weighting the graph edges by applying a Gaussian kernel to the Euclidean distance.

$$W_{ij} = P_{ij} \times e^{-d\left(x_i, x_j\right)/2\sigma^2}. \tag{8}$$

Then, define the graph density for an observation to be the sum of weights for all edges to the node representing the datapoint. Finally, normalize the sum weights by the total number of neighbors, as shown in Equation (9). The matrix is used to rank all samples according to their representativeness.

$$Gd(x_i) = \frac{\sum_i W_{ij}}{\sum_i P_{ij}}. \tag{9}$$

The query strategy based on the density of the graph reduces the weight of neighboring instances of the currently selected node x_i by reducing the weight of that node, as shown in Figure 4(c-4) and

$$Gd\left(x_j\right) = Gd\left(x_j\right) - Gd(x_i)P_{ij}. \tag{10}$$

This approach avoids both instances in the same dense domain being selected multiple times and UNSM falling into

Table 3: Performance of DMF classifier and primary classifiers.

Feature sets	Model	Accuracy	Precision	Recall	F_1 measure	FPR
F_{10}	DMF	**0.9793**	**0.9691**	**0.9901**	**0.9795**	**0.0315**
	RF	0.9783	0.9681	0.9891	0.9785	0.0325
	XGB	0.9777	0.9672	0.9890	0.9785	0.0335
	GNB	0.9778	0.9666	0.9898	0.9781	0.0341
F_{20}	DMF	0.9858	0.9764	**0.9956**	**0.9859**	**0.0210**
	RF	0.9858	0.9784	0.9935	0.9859	0.0218
	XGB	0.9856	0.9789	0.9926	0.9857	0.0214
	GNB	**0.9861**	**0.9792**	0.9933	0.9826	0.0210
F_{30}	DMF	**0.9872**	0.9774	**0.9975**	**0.9873**	0.0230
	RF	0.9859	0.9777	0.9945	0.9860	0.0226
	XGB	0.9867	0.978	0.9958	0.9868	**0.0223**
	GNB	0.9866	**0.9779**	0.9958	0.9867	0.0223
F_{40}	DMF	0.9895	0.9832	**0.9961**	**0.9896**	0.0170
	RF	**0.9896**	0.9830	0.9960	0.9896	0.0170
	XGB	0.9894	**0.9837**	0.9953	0.9895	**0.0164**
	GNB	0.9892	0.9833	0.9954	0.9893	0.0169
F_{50}	DMF	**0.9916**	**0.9858**	**0.9976**	**0.9917**	**0.0143**
	RF	0.9906	0.9853	0.9960	0.9906	0.0148
	XGB	0.9905	0.9850	0.9962	0.9906	0.0151
	GNB	0.9909	0.9854	0.9965	0.9909	0.0146

suboptimal queries in a certain direction. In practice, we set up a mechanism to prevent querying neighboring instances during the query process. When the query starts, we compare the next instance of the query with the k instances of the dense domain where the previous instance is located. If the same instance appears, this query is dropped. After the instance is removed from the pool, a new round of queries is performed.

4. Experiment and Results

In this section, we present the dataset, evaluation metrics, experimental setting, model results, and result analysis.

4.1. Dataset. We use the dataset of CTU-13 which is captured form a real-word network environment to verify the effectiveness of our AS-DMF framework [29]. The dataset was collected for over 24 hours long and consists of 100+ thousand encrypted traffic flows referring to 13 popular malware family. We filter 20,000 flows from benign traffic packet captured in a normal environment. There are over 100,000 flows to be extracted from malware packets, and 20,000 malicious flows are retained after filtering. The statistical information of dataset is shown in Table 1. We filter the pcap traffic packets by setting an *established* flag to extract the TLS flows that have established a full connection.

4.2. Experiment Setting

(1) *Metrics.* We evaluate the methods based on accuracy, precision, recall rate, F_1 measure, and false-positive rate (benign flows are predicted to be malicious flows, FPR). We also use mAcc (mean accuracy) to measure AS-DMF performance. The definitions are as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{11}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{12}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{13}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Reacll}}{\text{Precision} + \text{Recall}}, \tag{14}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{FN}}, \tag{15}$$

where TP (true positive) represents positive samples correctly identified as positive samples, FP (false positive) represents positive samples incorrectly identified as negative samples, FN (false negative) represents negative samples incorrectly identified as positive samples, and TN (true negative) represents negative samples correctly identified as negative samples.

(2) *Experimental environment.* This study built an experimental environment based on Python3.7 and ALiPy [30, 31]. The hardware device used for the experiments is a 64-bit Windows 10 operating

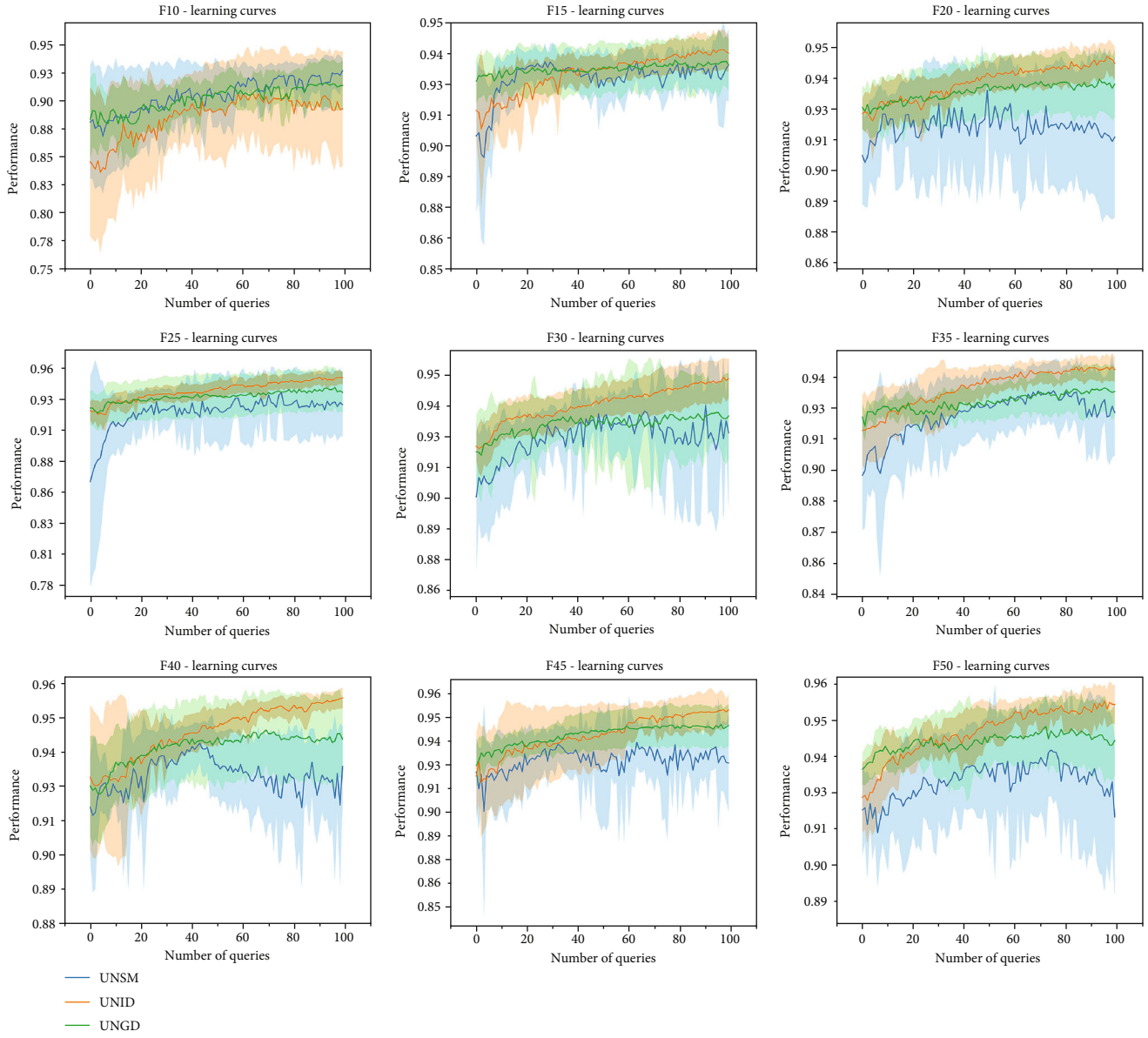

Figure 7: Query results of three strategies with different feature sets.

system with an Inter® Xeno® Gold 5210 CPU @2.20 GHz 2.19 GHz dual processors and 32 GB of RAM

(3) *Setting of parameters.* For the base learner of random forest, we use the trees with a number of 100, and the maximum depth of each tree is 12. For the base learner based on XGBoost, we choose the maximum depth of the tree to be 12 and the number of trees to be 120. Logistic regression model using "lbfgs" solver and *class_weight* set to "balanced" (dynamically adjusted weights). Moreover, we divide the sample set into a training set and a test set by 7 : 3 to avoid overfitting. We set the number of samples in $\mathscr{L}$ at 0.1% of the training set, and each query strategy queries 10 instances per round, for a total of 100 rounds of queries

(4) *Baseline.* We compare a series of state-of-the-art methods as baseline to evaluate the performance of our methods

The specific parameters of the baseline methods are as follows:

UNID: the exponent parameter $\beta = 1$, distance metric is Euclidean distance, and choosing information entropy as an uncertainty measure.

UNGD: The number of neighbors $k = 10$, and the distance metric is Euclidean distance.

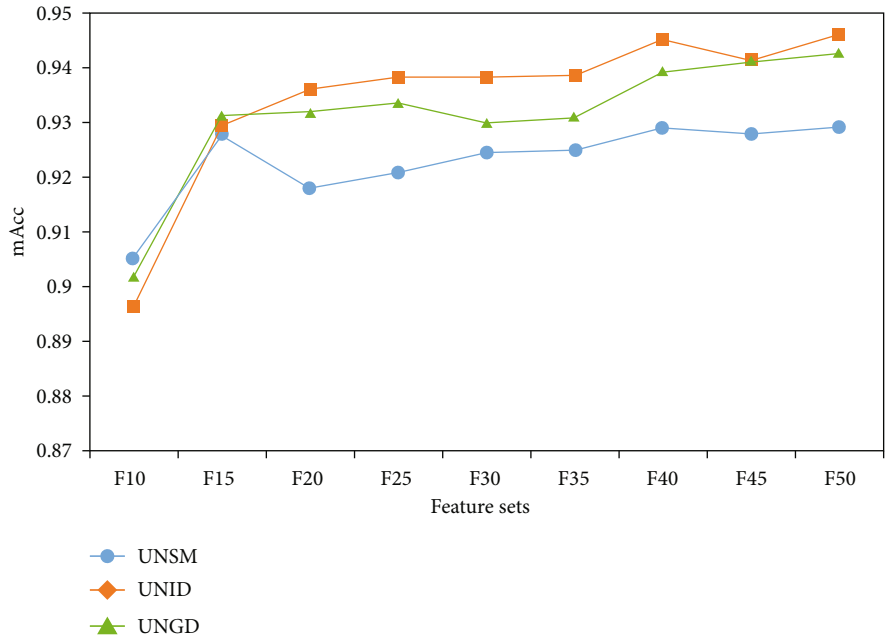

Figure 8: The final mean accuracy achieved by the 3 query strategies with different feature sets.

Table 4: Comparison of three query strategies in different feature set results (mAcc ± std).

Feature sets	UNSM	UNID	UNGD
F_{10}	0.9053 ± 0.0339	0.8965 ± 0.0491	0.9019 ± 0.0239
F_{15}	0.9276 ± 0.0141	0.9295 ± 0.0126	0.9314 ± 0.0102
F_{20}	0.9180 ± 0.0241	0.9359 ± 0.0103	0.9320 ± 0.0125
F_{25}	0.9211 ± 0.0317	0.9383 ± 0.0102	0.9336 ± 0.0169
F_{30}	0.9245 ± 0.0206	0.9383 ± 0.0106	0.9300 ± 0.0174
F_{35}	0.9250 ± 0.0209	0.9386 ± 0.0130	0.9311 ± 0.0093
F_{40}	**0.9292 ± 0.0174**	0.9451 ± 0.0159	0.9395 ± 0.0164
F_{45}	0.9277 ± 0.0208	0.9414 ± 0.0193	0.9411 ± 0.0109
F_{50}	0.9292 ± 0.0211	**0.9460 ± 0.0115**	**0.9427 ± 0.0112**

LAL: $n = 50$, $d = 5$, and the mode of data sampling is iterative, where n is the number of forest estimators and d is the forest depth.

ALEC: $r = 0.1$, and the block size threshold b is set to 3, where r is density radius, and blocks with $b < 3$ are no longer divided.

QUIRE: kernel = 'linear', $\lambda = 1$, where λ is a regularization parameter used in the regularization learning framework.

4.3. Experiment

(1) *Feature selection results.* After extracting features from the dataset, a 98-dimension feature set is obtained. We use the feature selection mechanism described in Section 3.3 to rank these features by MIC values. In practice, 10 sets of experiments are designed for comparison on the full feature set (F_{all}) and the postselection feature set F_m ($m \in [10, 50]$), respectively, to verify the effectiveness of the feature selection mechanism. The comparison results are shown in Table 2 and Figures 5 and 6, and the values in bold represent the best results

(2) *DMF classifier.* To examine the performance of the DMF model, we compared the DMF model with the three primary classifiers on the F_m ($m \in \{10,20,30,40,50\}$). As shown in Table 3, RF, XGB, and GNB represent random forest classifier, XGBoost classifier, and Gaussian Naive Bayes classifier, respectively

Table 5: Comparison results of each query strategy (mAcc ± std).

Feature sets	UNID	UNGD	QUIRE	LAL	ALEC
F_{10}	0.8965 ± 0.0491	0.9019 ± 0.0239	0.8282 ± 0.0851	**0.9026 ± 0.0322**	0.7394 ± 0.0163
F_{20}	**0.9359 ± 0.0103**	0.9320 ± 0.0125	0.8944 ± 0.0418	0.9339 ± 0.0113	0.9173 ± 0.0053
F_{30}	**0.9383 ± 0.0106**	0.9300 ± 0.0174	0.9118 ± 0.0188	0.9364 ± 0.0103	0.9187 ± 0.0052
F_{40}	**0.9451 ± 0.0159**	0.9395 ± 0.0164	0.9256 ± 0.0173	0.9443 ± 0.0560	0.9189 ± 0.0008
F_{50}	**0.9460 ± 0.0115**	0.9427 ± 0.0112	0.9245 ± 0.0103	0.9455 ± 0.0050	0.9184 ± 0.0010

(3) *AS-DMF results*: an experiment with 9 groups are designed by using the reduction feature sets to observe the performance of AS-DMF lightweight framework. The query strategy includes uncertainty sampling strategy with entropy method, uncertainty sampling with information density (UNID), and uncertainty sampling with graph density (UNGD). The results of the query are shown in Figures 7 and 8 and Table 4, and std represents the overall standard deviation

(4) *Comparison with advanced methods*. We compared density-based methods with LAL, ALEC, and QUIRE mentioned in Section 2.2. The comparison results are shown in Table 5

4.4. Analysis of Feature Selection Results and DMF Classifier. The comparison results are shown in Tables 2 and 3. We can obtain the following conclusions.

(1) *Analysis of feature selection results.* The DMF classifier achieves the best performance in the fall feature set, as expected, and outperforms other feature sets in most cases, as is shown in Table 2. However, when the target feature set is F_{10}, the accuracy, precision, recall, and F_1 measure of the DMF classifier differ by only 0.0156, 0.0226, 0.080, and 0.0154, respectively, and can be trained faster. From the aspect of the informativeness of feature sets, the differences between the various performance metrics are small, indicating that F_{10} feature set and F_{all} feature set similar of identical distribution and also indicating that the feature selection mechanism of this paper is effective

As the dimensionality of the feature set increase, the similarity of informativeness becomes larger, which is expressed in the decreasing various performance metrics. And the FPR of the DMF classifier, as shown in Figure 6, shows a decreasing trend, which means fewer misclassified samples.

(2) *Analysis of DMF classifier.* Comparing them with different sequences, as is shown in Table 3, the DMF classifier always outperforms the primary learners in recall rate and F_1 measure. And it can obtain the best performance on most of the metrics. However, the random forest classifier and Gaussian Naive Bayes classifier outperformed the DMF classifier by precision on F_{20}, F_{30}, and F_{40}. And the XGBoost classifier shows better performance in FPR, which outperforms the DMF classifier by 0.0003 and 0.0006 in F_{20} and F_{30}, respectively. The reason for the degraded performance of the DMF classifier is that, compared to average the probability values of each single component, the DMF classifier chooses a logistic regression model as a secondary learner to dynamically adjust the weights of the primary learners. Compared to single primary learners, DMF classifier has a better generalization ability. In general, the DMF classifier has good classification performance and outperforms a single primary learner

4.5. Analysis of AS-DMF Results. The experimental results of AS-DMF framework are shown in Figures 7 and 8 and Tables 4 and 5, and we can draw some conclusions.

(1) AS-DMF framework can indeed reduce the complexity of the detection process and improve data utilization. When comparing performance with different feature sets, the F_{50} outperforms the others with more than 0.0046 in mAcc. With feature selection mechanism, the feature sets selected from the raw encrypted traffic are guided by the ANOVA and MIC to store richer information. Furthermore, the performance of the DMF classifier improves as the informativeness of the feature set increases

(2) The performance of the three query strategies on a specific feature set steadily improves with the number of queries, but the UNSM is more volatile, such as F_{20}, F_{40}, and F_{50}. In most cases, the strategy combined with density works better than the UNSM alone. Under contract, the volatility of UNID is minimal, and the performance of DMF can be improved smoothly using this query strategy. The reason is that the UNID strategy combines global uncertainty and information density, which makes it possible to choose instances that are more informative and representative

(3) The density-based query strategy performs better than UNSM. Although using the same input (e.g., $\mathscr{L}$, feature set, and classifier), the UNID and UNGD outperform the UNSM in Figure 8, because the density-based query strategy can search the most informative and representative instances for DMF classifier to train. Therefore, our strategy allows DMF classifier to learn the most knowledge with the least training cost

(4) The higher the dimensionality of the feature set, the fewer query numbers are required to achieve the preset mAcc value. This phenomenon shows that the informativeness and representativeness contained in the instance itself affects the query results. In general, the higher the performance achieved by the more informative feature set for the same query cost, but not absolutely, e.g., F_{40} under UNSM. In practice, it is necessary to balance the size of the feature set and the number of queries

(5) Density-based methods has overall higher mAcc values than other methods, as shown in Table 5, when setting the same number of queries. As the dimensionality of the feature set increases ($10 < m \leq 50$), UNID outperforms all advanced algorithms in mAcc and stability. Therefore, the density-based methods we proposed has better superiority

5. Conclusion

In this paper, we design a lightweight framework for TLS encrypted malicious traffic detection named AS-DMF. It selects the representative features by feature selection mechanism. The AS-DMF takes an uncertainty sampling and density-based query strategy to search the most informative and representative instances of the unlabeled samples and train the DMF classifier with a pool-based active learning approach. We evaluate our methods on feature sets of different dimensions and draw the following conclusions: (1) the feature selection mechanism effectively reduces the feature complexity and the computational overhead of the DMF model; (2) UNID query strategy avoids getting into suboptimal queries by combining global uncertainty and local information density; (3) UNGD strategy combines global uncertainty and graph density to move queries in the direction of higher density; (4) both the UNID and UNGD query strategies can query for informative and representative instances; (5) AS-DMF framework enables DMF classifier to achieve lightweight detection at both feature and data levels with 1/40 labeling cost and less than 5% mAcc loss. In the future, we will investigate active learning lightweight methods under unbalanced data.

Data Availability

The data that support the findings of this study are openly available in CTU-13 at https://www.stratosphereips.org/datasets-ctu13.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (no. 2016YFC0801800).

References

[1] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of TLS (without decryption)," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 3, pp. 195–211, 2018.

[2] S. Gallagher, *Nearly Half of Malware Now Use TLS to Conceal Communications*, Technical Report, 2021.

[3] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1723–1732, Halifax, NS, Canada, 2017.

[4] T. Radivilova, L. Kirichenko, D. Ageyev, M. Tawalbeh, and V. Bulakh, "Decrypting SSL-TLS traffic for hidden threats detection," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pp. 143–146, Kyiv, Ukraine, May, 2018.

[5] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "Deep-full-range: a deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[6] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–35, 2021.

[7] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: a flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1171–1179, Paris, France, June 2019.

[8] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[9] Y. Yang, C. Kang, G. Gou, Z. Li, and G. Xiong, "TLS/SSL encrypted traffic classification with autoencoder and convolutional neural network," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 362–369, Exeter, UK, January 2019.

[10] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, "Hunting malicious TLS certificates with deep neural networks," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (AISec '18)*, pp. 64–73, Toronto, Canada, October, 2018.

[11] V. L. Nguyen, M. H. Shaker, and E. Hullermeier, "How to measure uncertainty in uncertainty sampling for active learning," *Machine Learning*, vol. 111, no. 1, pp. 89–122, 2022.

[12] R. Dai, C. Gao, B. Lang, L. Yang, H. Liu, and S. Chen, "SSL malicious traffic detection based on multi-view features," in *Proceedings of the 2019 the 9th International Conference on*

Communication and Network Security, pp. 40–46, Chongqing, China, November, 2019.

[13] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, pp. 35–46, Vienna, Austria, October, 2016.

[14] K. C. Pai, S. Mitra, and M. Chari, "Novel TLS signature extraction for malware detection," in *Proceedings of the 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–3, Bangalore, India, July, 2020.

[15] J. Guo, Y. Sang, P. Chang, X. Xu, and Y. Zhang, "MGEL: a robust malware encrypted traffic detection method based on ensemble learning with multi-grained features," *Computational Science - ICCS*, vol. 12743, pp. 195–208, 2021.

[16] H. Yang, Q. He, Z. Liu, and Q. Zhang, "Malicious encryption traffic detection based on NLP," *Security and Communication Networks*, vol. 2021, Article ID 9960822, no10 pages, 2021.

[17] T. Yu, F. Zou, L. Li, and P. Yi, "An encrypted malicious traffic detection system based on neural network," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 62–70, Guilin, China, October, 2019.

[18] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, article 102419, 2020.

[19] X. Q. Zhang, M. Zhao, J. Y. Wang, S. Li, Y. Zhou, and S. Zhu, "Deep-forest-based encrypted malicious traffic detection," *Electronics*, vol. 11, no. 7, p. 977, 2022.

[20] Y. C. Chen, Y. J. Li, A. Tseng, and T. Lin, "Deep learning for malicious flow detection," in *2017 IEEE 28th Annual International Symposium on Personal* of *Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, Montreal, QC, Canada, October, 2017.

[21] M. Fang, Y. Li, and T. Cohn, *Learning How to Active Learn: A Deep Reinforcement Learning Approach*, EMNLP, 2017.

[22] P. Liu, L. Wang, R. Ranjan, G. He, and L. Zhao, "A survey on active deep learning: from model driven to data driven," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–34, 2022.

[23] S. Ebert, M. Fritz, and B. Schiele, "ALF: a reinforced active learning formulation for object class recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3626–3633, Providence, RI, USA, June 2012.

[24] D. Zha, K. H. Lai, M. Wan, and X. Hu, "Meta-AAD: active anomaly detection with deep reinforcement learning," in *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 771–780, Sorrento, Italy, November, 2020.

[25] M. Wang, F. Min, Z. H. Zhang, and Y. X. Wu, "Active learning through density clustering," *Expert Systems with Applications*, vol. 85, pp. 305–317, 2017.

[26] Y. Shi, Z. Yu, W. Cao, C. L. P. Chen, H. S. Wong, and G. Han, "Fast and effective active clustering ensemble based on density peak," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3593–3607, 2021.

[27] K. Konyushkova, S. Raphael, and F. Pascal, "Learning active learning from data," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[28] S. J. Huang, R. Jin, and Z. H. Zhou, "Active learning by querying informative and representative examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 1936–1949, 2014.

[29] *Stratosphere Laboratory Datasets*, Stratosphere, 2015.

[30] F. Pedregosa, G. Varoquaux, A. G. Alexandre et al., "Scikit-learn: machine learning in Python," *The Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[31] Y.-P. Tang, G.-X. Li, and S.-J. Huang, "ALiPy: active learning in Python," 2019, https://arxiv.org/abs/1901.03802.