CQNet: A clustering-based quadruplet network for Decentralized Application classification via encrypted traffic

Yu Wang[1,2], Gang Xiong[1,2], Chang Liu[1,2], Zhen Li[1,2], Mingxin Cui[1,2], and Gaopeng Gou[1,2(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
{wangyu1996, xionggang, liuchang, lizhen, cuimingxin, gougaopeng}@iie.ac.cn

Abstract. Decentralized applications (DApps), along with the development of blockchain technology, are increasingly developed and deployed on blockchain platforms. DApps based on the same platform usually adopt similar traffic encryption settings and the same communication interface, leading to traffic less distinguishable. However, existing classification methods either require manual-design features or need lots of data to train the classifier, otherwise suffering from low accuracy. In this paper, we apply metric learning to DApps encrypted traffic classification problem and propose the clustering-based quadruplet network (CQNet). The CQNet can filter out useless samples to reduce the training dataset's redundancy data by utilizing the proposed algorithm, thereby improving the classifier's efficiency. Moreover, we adopt a quadruplet structure that can mine more restrictive relationships among quadruplets and provide rich information to the classifier. Our comprehensive experiments on the real-world dataset covering 60 DApps indicate that CQNet can achieve excellent performance with high efficiency and is superior to the state-of-the-art methods in terms of accuracy and efficiency.

Keywords: Encrypted traffic classification · Decentralized applications · Clustering · Quadruplet networks · Feature embedding · Metric learning.

1 Introduction

With the surge in popularity and rapid development of blockchain, the volume of decentralized applications (DApps) rises sharply. That is attributed dramatically to DApps' resistance to censorship, making them more freedom. Until now, more than 3700 DApps are deployed on different blockchain platforms, such as Etherem (81.91%), Eos (8.97%), Steem (1.62%), etc. We focus on Ethereum [1] DApps in this paper, as it also has the most significant number of daily active users, about one hundred thousand [2]. DApps are autonomously managed without the control of a single entity and blockchain technology can naturally provide anonymity to each user. These are the unique advantages of DApps that traditional applications cannot provide.

However, the problems faced by DApps and centralized applications are similar, i.e., how to manage the DApps network better and ensure a secure network environment. Network traffic classification naturally arises because it is a vital task for these two problems [10]. According to different priority strategies, DApps traffic should be classified for better network management. Malicious DApps traffic should also be identified for anomaly detection, thus guaranteeing the DApps network security. Traffic classification attracts many researchers and lots of methods have been proposed for website classification [3,14,15,19,20], mobile application classification [6,12,21,22,26] and user behavior classification [4,24], but few efforts have been made on DApps encrypted traffic classification.

While prior work has achieved good accuracy results, these methods designed the sophisticated network architecture combined with features extracted artificially from flows [9,10,15,18,20], which are based on professional knowledge, human effort and time-cost. Some studies use fewer or simpler features to achieve good results [4,5,11,21,24], which are unsatisfactory on DApps encrypted traffic classification. Some studies also resort to large dataset to improve performance (e.g., training on a large dataset contains 956 thousand flows [10]), which leads to the redundancy problem of dataset. All in all, the task of DApps encrypted traffic classification can be decomposed into two subtasks. The first is how to let the network automatically extract features and accurately classify DApps traffic. The second is how to improve the efficiency of training and testing.

In this paper, We propose a novel classification model using metric learning named clustering-based quadruplet network (CQNet) for DApps encrypted traffic classification. The CQNet aims to learn an embedding space, thereby mapping each encrypted traffic flow into the metric space to form an embedded vector. CQNet includes two mechanisms, filtration of easy dataset (FE-set) algorithm and quadruplet network. FE-set algorithm combines the three parts of mini-batch KMeans, Kuhn-Munkres algorithm and exploring centers of clusters to filter out easy samples from all flows. Hence, the original dataset can be split into the easy dataset and hard dataset. We construct quadruplets on the hard dataset as the quadruplet network's input, which can leverage more constraints among samples. FE-set algorithm and the quadruplet network are designed to solve DApps encrypted traffic classification efficiency and accuracy, respectively.

Contributions: Our contributions can be summarized as follows:

1) We explored the redundancy problem of DApps encrypted traffic dataset for the first time. FE-set algorithm can divide easy dataset and hard dataset, thereby improving the training efficiency.
2) We designed a quadruplet network, which can leverage more restrictive relationships among quadruplets and provide rich information to the classifier.
3) Our CQNet achieves outstanding results on the real-world network traffic data for the DApps encrypted traffic classification and outperforms the state-of-the-art methods.

Roadmap. Sec. 2 summarizes the related work. Sec. 3 describes the preliminaries. The detailed system architecture is proposed in Sec. 4. Sec. 5 shows the evaluation results. Finally, we conclude this paper in Sec. 6.

2 Related Work

Many conventional traffic classification methods are no longer suitable for current encryption scenarios, such as DPI (Deep Packet Inspection) and port-based methods. Here, we only introduce studies that are closely related to our work. Hence, prior work on encrypted traffic classification falls into three broad categories: (1) web application classification, (2) mobile application classification and (3) decentralized applications classification.

2.1 Web Application Classification

Panchenko et al. [13] presented a website fingerprinting method at Internet Scale. The accumulated sum of packet sizes was used to represent the progress of webpage loading. Hayes et al. [7] proposed a robust website fingerprinting technique based on Random Forest. The leaves of Random Forest are used to encode a new representation of the websites, thereby transforming 175 features into a different feature space, which are fed to a KNN classifier. Shi et al. [19] introduce an efficient feature optimization approach, based on a deep learning framework. It enhanced traffic classification performances by removing the redundant features. [20] designed convolutional Neural Networks (CNN) with sophisticated architecture. The model can have high accuracy for Tor websites identification by using only the packet direction sequence.

2.2 Mobile Application Classification

In [26], the authors used a suite of inference techniques to reveal a specific user action (i.e., send a tweet) on the Twitter app installed on an Android smartphone. Taylor et al. proposed a robust application identification method with the concept of the burst. They used statistical features of packet length with the Random Forest classifier to build the Appscanner [21], which can identify 110 applications with 96% accuracy. Condi et al. clustered the streams of each application user behavior by clustering methods [4], then they calculated the dynamic warping distance for each flow in terms of packet length. Several studies applied Markov models to identify smartphone applications [8, 17].

2.3 Decentralized Applications Classification

DApps are emerging as a new service paradigm that relies on the underlying blockchain platform. Shen et al. generated high-dimensional features by fusing time series, packet length and burst sequence [18]. The accuracy of DApps traffic classification reached 90%. But the training and testing time of the method is much longer than other methods because of the large input vectors. Wang et al. [24] found that more than 60% of flows are short flows, leading to the burst feature's poor performance, so they only extracted time series and packet length to build a classifier using Random Forest. In this paper, we attempt to design a new deep learning network structure without needing professional knowledge and hand-crafted features.

3 Preliminaries

While prior work of encrypted traffic classification has obtained some insights, encrypted traffic classification in distributed DApps is still a considerable challenge. In this section, we provide DApps background, give the definition of DApps encrypted traffic classification problem and the limitation of existing methods.

3.1 DApps Background

DApps consist of frontend and backend, where the frontend implements user interfaces uniformly defined by blockchain platforms to present pages. Smart contracts acting as the backend connect to blockchain networks, utterly different from the backend of the web application. Smart contracts are the functionalities that determine the results of operations in DApp performed by users.

Most DApps provide browser plug-ins or websites so that users can easily access. When a user visits a DApp, the DApp client obtains the server's IP address equipped with the corresponding smart contract through **DNS** query and sends a request. Then, the smart contract determines the operation results and sends them to the **mempool**. All data records of these are packaged in the form of blocks by miners and stored on the distributed ledger. After these steps, the client obtains the updated information from blockchain to present pages through a new list of servers acquired from the smart contract server.

Ethereum DApps adopt the same communication interface, employ similar traffic encryption settings, store data and run smart contracts in the same blockchain platform, making the encrypted traffic of DApps more challenging to distinguish than the classification of traditional applications.

3.2 Problem Definition

The DApps encrypted traffic classification task is to classify flows into specific DApps with the raw traffic data as the only classification information. Assume that there are N samples and K classes of DApps in total. The z-th sample is defined as $f_z = \left[b_1^{(z)}, \ldots, b_{n_z}^{(z)}\right]$, where n_z is the length of f_z and $b_i^{(z)}$ stands for the size of packet $b_i^{(z)} \in [0, 1500)$ in bytes at time step i. The DApps label of f_z is denoted as L_z, $1 \leq L_z \leq K$. Our goal is to build a model to predict a label $\hat{L}_z$ that is exactly the real label L_z.

3.3 Limitation of Existing Methods

Existing encryption traffic classification methods face two challenges, efficiency (RQ1) and accuracy (RQ2).

Efficiency To label dataset, the traffic classification is different from other research fields (i.e., **CV**, **NLP**). Traffic classification dataset can be automatically labeled by SNI (Server Name Indication) or process ID. Therefore, the

dataset may contain many samples (e.g., nearly one million flows in [10]), including easy, semi-hard and hard samples [16]. Since the overly easy samples can satisfy the constraints well and produce almost zero loss, they do not contribute to the parameter update during back-propagation and consume a lot of time. So, how to filter out easy samples from the dataset to improve training efficiency? (**RQ1**)

Accuracy As introduced in Sec. 2, the premise of good classification performance should be that no experts or a simple model to use.

We are inspired by the triplet network [16]. Flows could be placed in a common metric space and the explicit closeness could be measured by Euclidean distance. The function for each pair of flows interaction can be defined as:

$$d(f_a, f_p) = \left\| \boldsymbol{\alpha}_{f_a} - \boldsymbol{\alpha}_{f_p} \right\|_2^2 \tag{1}$$

where $\| * \|_2$ denotes the L_2 - norm. α_f is the representation of traffic flow in the low dimensional space.

The triplet loss: Given a triple (f_a, f_p, f_n), $1 \leq a, p, n \leq N$, the label $L_a = L_p \neq L_n$. The loss guarantees the distance between anchor f_a and negative f_n is larger than that between f_a and positive f_p by a fixed margin:

$$d(f_a, f_p) + m \leq d\left(f_a, f_n\right) \tag{2}$$

The final objective function could be expressed as following:

$$\mathcal{L} = \sum_{L_a = L_p} \sum_{L_a \neq L_n} \left[d(f_a, f_p) - d\left(f_a, f_n\right) + m \right]_+ \tag{3}$$

where $[x]_+ = max(x, 0)$ represents the hinge loss.

Fig. 1: An illustration of metric learning by using triplet network. The same color indicates that the samples belong to the same DApps. After training, the sample with blue color is close to the yellow one and far from the other blue.

From a geometric perspective, the triplet loss only considers the relationship of two edges. The negative flow's position may be far from the same DApp's flow and close to another DApp's flow. This undoubtedly compromises the performance of the classifier. As shown in Fig. 1, the learning process will be stopped following the constraint of Equation 2. We can observe that the sample with blue color is close to the yellow one and far from the other blue.

Due to the different distribution of flows for different DApps, the fixed margin for all DApps may not perform well. So, how to utilize more relationships among samples and adjust the margin to train the classifier? (**RQ2**)

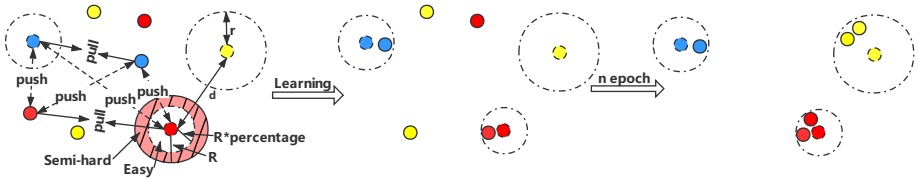


Fig. 2: An illustration of our proposed method CQNet. R represents each class's initial radius after clustering (including easy samples and semi-hard samples), R*percentage is the range that we finally selected as the easy dataset. And the semi-hard samples are classified into the hard dataset. The right of the figure represents the final situation after completed training.

4 CQNet

In this section, we will illustrate our clustering-based quadruplet network (CQNet) to circumvent the above two issues (RQ1 and RQ2). An illustration of CQNet is shown in Fig. 2. The CQNet aims to learn embeddings, such that flows generated from distinct DApps are embedded into a common metric space. Thus the embedded features favor discriminative between different DApps. The overall architecture of CQNet is shown in Fig. 3. We outline the underlying ideas of the work, including FE-set, the proposed quadruplet network.

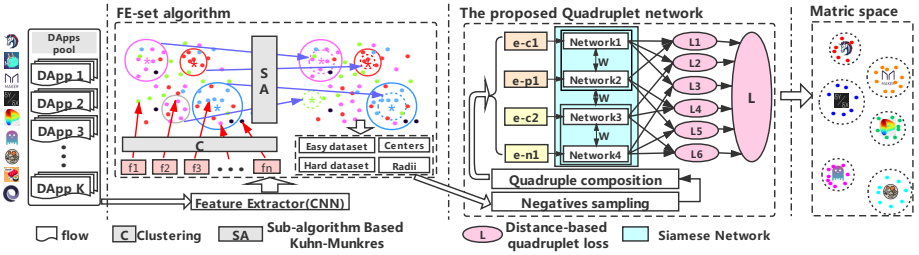
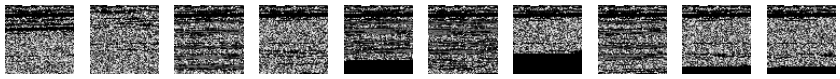


Fig. 3: The architecture overview of the CQNet. The FE-set algorithm aims to divide easy dataset and hard dataset, thus improving training efficiency. The quadruplet network aims to find constraints to improve classification accuracy.

4.1 FE-set (RQ1)

According to the problem definition in Sec. 3.2, given a sample f_z. In order to extract features automatically, we utilize the first n bytes (n = 64*64) to represent the flow, the raw data is $f_z = \left[b_1^{(z)}, \ldots, b_{m_z}^{(z)}\right]$, $\sum_{i=1}^{m_z} b_i^{(z)} = n$. For the whole dataset, $F = [f_1, \ldots, f_N]$. As shown in Fig. 4, F are converted to gray

images for visualization [23]. By using the CNN architecture model pre-trained on the fashion-mnist, F is transformed into the embedded feature as the input to filter out easy samples, which are defined as $X = [x_1, \ldots, x_N]$.



(a) Meg (b) Avx (c) Bfh (d) Dou (e) Tok (f) For (g) 1in (h) Cha (i) Axi (j) Ael

Fig. 4: Visualization of some classes of DApps traffic. It is obvious that DApps are different from each other and only a few images are similar.

The proposed algorithm. Considering the task of screening easy samples from N flows to form the easy dataset, $E = [EF_1, \ldots, EF_K]$, Where $EF_i = \left[F_1^{(i)}, \ldots, F_{l_i}^{(i)}\right]$, EF_{l_i} represents the easy samples belonging to the label i and l_i means the number of easy samples with label i, which may be zero. And the hard (including semi-hard and hard) dataset, $H = [HF_1, \ldots, HF_K]$. FE-set algorithm has the main idea: when samples belonging to the same class are also predicted to be the same class after using clustering, these samples are relatively close in the embedding space and form a cluster. In other words, these samples are easy to distinguish. So we choose a distance-based algorithm (mini-batch k-means) instead of other clustering algorithms (i.e., density-based methods). Next, we formally present the process, as exhibited in Algorithm 1.

FE-set algorithm considers the embedded features $X = [x_1, \ldots, x_N]$ as the input and starts with an initialization of the easy dataset E and hard dataset H. EX_i and HX_j represents the embedded features of each class. Then, it clusters X into K clusters through the mini-batch K-Means clustering algorithm. The number of cluster K is usually specified according to the category of DApps. Each flow in the dataset has a real label L_z and a predicted label $\hat{L}_z$. To correspond the real labels L and predicted labels $\hat{L}$ in the maximum number, we transform the task into a weighted bipartite graph maximum matching task, **Kuhn-Munkres** algorithm is used to solve this problem. We construct a matrix R as the input, the abscissa is the real label, the ordinate is the predicted label, the value in the matrix can be understood as the weight value and the number of samples that satisfy $R(i,j), L_{(i,j)} = i, \hat{L}_{(i,j)} = j$. The real label corresponds to the predicted label one-to-one after utilizing the **Kuhn-Munkres** algorithm. The samples that corresponded successfully are screened from the dataset and put into the easy dataset E. The remained samples are put into the hard dataset H.

Because we use the maximum matching, i.e., the amount of samples contained in the corresponding labels should be the largest, so the number of samples contained in a few correspondences between the true and predicted label may be 0 (i.e., $R(i,j) = 0$). These labels are added into NL and the remained labels are added into YL. Then, we directly use K-Means algorithm to obtain the center for each class in E and construct the center list $C = [c_1, \ldots, c_K]$, there may be

Algorithm 1 Filtration of easy dataset (FE-set).

Input:
the embedded features $X = [x_1, \ldots, x_N]$, the real label $L = [l_1, \ldots, l_{N`14r}]$,
the raw data $F = [f_1, \ldots, f_N]$, the number of clusters K.
Output:
the easy dataset $E = [EF_1, \ldots, EF_K]$, the hard dataset $H = [HF_1, \ldots, HF_K]$,
the center set $CF = [cf_1, \ldots, cf_K]$, the radius set $Perc\_D = [perc\_d_1, \ldots, perc\_d_K]$.
Initialize E, H as empty sets
Predict clusters $\hat{L} = MiniBatchKMeans(X, K)$
Kuhn-Munkres($R = (L, \hat{L})$), obtain the corresponding relationship RE
Separate F into E and H according to RE
for $i \in [1, K]$ **do**
  if $len(EF_i) > 1$ **then**
    put i into a list YL
  else
    put i into a list NL
  end if
end for
for $i \in YL$ **do**
  put center $c_i = KMeans(EX_i, 1)$ into C, put f_{c_i} into CF
  for $X^i_{l_i} \in EX_i$ **do**
    put $d(X^i_{l_i}, c_i)$ into a list D
  end for
  Sort D, put the n percentile of D into $Perc\_D$
  Keep the ρ percentage closest samples in E. Put the others into H
end for
for $j \in NL$ **do**
  for $X^j_m \in HX_j$ **do**
    for $i \in YL$ **do**
      put $d_i(X^j_m, c_i)$ into a list $DC = [dc_1, \ldots, dc_K]$
    end for
    obtain $(index\_min, d\_min) = min(DC > 0)$, let $do_{index\_min} = d\_min$ for the list $DO = [do_1, \ldots, do_K]$
  end for
  obtain $(index, index\_min, d) = max(DO)$, put X^j_{index} into C, put f^j_{index} into CF, let $nl\_dj = d$ for the list NL_D
  calculate $perc\_d_j = (nl\_d_j - perc\_d_{index\_min}) * \rho$
end for
return $E, H, CF, Perc\_D$

$c_i = 0$. However, considering the samples at the edge of clusters cannot always be accurately predicted when clustering, we calculate the Euclidean distance D between each sample in the class to the center and sort D (The most prolonged d is R in Fig. 2). The percentage ρ closest samples to the center are selected as the real easy samples (i.e., r or R*percentage in Fig. 2). We record the n percentile of D as $perc\_d_i$ and use it as the radius of category i. The radius set is $Perc\_D = [perc\_d_1, \ldots, perc\_d_K]$. The rest of the samples can be regarded as semi-hard samples and transferred to H, so we obtain the two sets, E, H.

From the above, we get a set of centers C (the value of $c_i \neq 0$ if $L_i \in YL$). However, not all categories have a center in C ($L_j \in NL$), so we need to find the imaginary center of samples whose labels are in NL. For $L_j \in NL$, it calculates the Euclidean distance between each sample in category L_j and other existing centers ($c_i, i \in YL$), takes the smallest value, then gets the sample with the maximum distance (denoted as $NL-D$) from the set formed by all the minimum values. This sample is used as the imaginary center of category L_j, and added into C. We can also obtain CF. The label of the nearest center and the distance

are recorded as L_q and $nl\_d_{L_j}$, respectively. But there is no suitable radius to represent category L_j, thereby calculating $perc\_d_{L_j}$ as follows:

$$perc\_d_{L_j} = \left(nl\_d_{L_j} - perc\_d_{L_q}\right) * \rho \tag{4}$$

4.2 The Proposed Quadruplet Network (RQ2)

The triplet network has two constraints, the rationale is to favor distances between the anchor and negative sample while penalizing distances between the anchor and positive sample, which is also our network's goal.

Since each category's easy samples are around the corresponding center, the encryption traffic classification problem is turned into calculating the distance between the samples and the centers. After training the hard samples through the model, all samples should be close to each cluster's center. For testing, we can calculate the distance between the sample and each center and the category of the closest center is the class of the sample. In this subsection, we will introduce our proposed network, which leverages more information among samples to improve classification accuracy.

Quadruplet composition. With Algorithm 1, we have filtered out easy samples, thereby reducing the network's training pressure. Inspired by the center loss [25], choosing the cluster center as the anchor for training is better and more efficient than randomly choosing two positive samples. However, only using the center loss often cannot achieve good results and it needs to be combined with other loss functions to obtain more feedback information.

We define a quadruplet, which adds a negative sample compared to the triplet. The illustration of CQNet is shown in Fig. 2. The positive and negative samples and the two anchors are embedded into the same metric space such that samples relevant to the two anchors form clusters around them. The main idea is to measure the six edges' relationships marked with 'Pull' and 'Push'. Given two different classes of center from CF as two anchors, $(cf_i, y_i), (cf_j, y_j)$ and two hard samples corresponding to respective center classes, $(f_i^{(y_i)}, y_i)$ (short for (f_i, y_i)), $(f_j^{(y_j)}, y_j)$ (short for (f_j, y_j)). Besides the two constraints from the triplet network, several constraints can be obtained from pairwise combinations within the quadruplet:

1) f_j should be close to cf_j.
2) f_i(or f_j) should be far away from cf_j(or cf_i).
3) cf_i should be far away from cf_j.

Negatives sampling. As for the quadruplet, some classes of negative samples may be far away from the positive sample, random sampling from all classes may lead to many easy restrictive relationships. FE-set algorithm 1 aims to filter out simple samples in positive samples, and the purpose of negatives sampling is to filter out easy negative samples.

For any two centers (c_i, c_j), we calculate the Euclidean distance $d(c_i, c_j)$, put it into the matrix CR, the abscissa and the ordinate both represent labels. After selecting f_i, we get the top τ ($\tau = 20$ in this paper) categories closest to y_i through the y_i-th row of CR and f_m is randomly selected from these categories to train the quadruplet network more effectively.

Network Structure To accommodate the set of quadruplets, we propose a quadruplet network, which can transform f_i to z_i with a nonlinear mapping $Q_\theta : F \to Z$, θ is the set of weight parameters, F is the raw data of traffic flows. Since the quadruplet is from the same domain, the quadruplet network comprises two Siamese networks, the weights shared. We define $\phi(\boldsymbol{z}_i, \boldsymbol{z}_j) = d(z_i, z_j)$ as the similarity function between f_i and f_j.

Distance-based quadruplet loss The loss function often sets the margin to a fixed value such as 1 or 2. However, the flow distribution of each category is different. According to $Perc\_D$, each cluster's radius is a different value in our dataset, $r \in (0.5, 3)$. And the distance between each center pair is also different. So the proposed distance-based quadruplet loss is improved based on Equation 3.

As shown in Fig. 2, given outputs quadruplet features $\left[z_f^i, z_f^j, z_{cf}^i, z_{cf}^j\right]$, we can obtain six pairwise similarity value as $\left\{\phi\left(z_n^m, z_{n'}^{m'}\right)\right\}_{n,n' \in \{cf,f\}}^{m,m' \in \{y_i,y_j\}}$. For each pair, there are two states, pulling close or pushing away. However, there is a restrictive relationship, cf_i should be far away from cf_j, our overall method is related to the cluster center and radius so that we discard this restrictive relationship. Hence, the quadruplet loss with the fixed margin (i.e., m for pulling close and m' for pushing away) can be denoted as follows:

$$\begin{aligned} L_{m,m'} = &\left[-d(z_f^i, z_f^j) + m'\right]_+ + \left[d(z_f^i, z_{cf}^i) - m\right]_+ + \left[d(z_f^j, z_{cf}^j) - m\right]_+ \\ &+ \left[-d(z_f^i, z_{cf}^j) + m'\right]_+ + \left[-d(z_f^j, z_{cf}^i) + m'\right]_+ \end{aligned} \tag{5}$$

We have obtained clusters' radii (i.e., $Perc\_D$) through Algorithm 1 and the Euclidean distance between any two centers (i.e., CR). As shown in Fig. 2, $r \in Perc\_D$ and $d \in CR$. We adaptively set the margin threshold to make the network accommodate better to the distribution of different categories.

In order to limit the sample to the cluster, given $m(f_i, cf_i) = perc\_d_{y_i}$ and $m(f_j, cf_j) = perc\_d_{y_j}$. For the margin of sample and other centers, the negative sample f_j should be outside of the ball with $d(cf_i, cf_j) - r(y_j)$, so $m(f_i, cf_j) = CR(y_i, y_j) - perc\_d_{y_j}$ and $m(f_j, cf_i) = CR(y_i, y_j) - perc\_d_{y_i}$. For the margin of two samples with different classes, the distance between them should be larger than the distance between the closest samples of the two clusters, so $m(f_i, f_j) = CR(y_i, y_j) - perc\_d_{y_i} - perc\_d_{y_j}$.

5 Performance Evaluation

In this section, we first introduce the dataset and experimental settings. Then, we conduct extensive experiments to evaluate the effectiveness of CQNet, including hyperparameters selection, performance comparison and ablation studies.

5.1 Dataset Collection

We capture the traffic flows of DApps through the routers of a laboratory, meanwhile filter the non-SSL/TLS encrypted traffic. Due to **Chrome** is used as the designated browser by some DApps, so all visits use **Chrome**. The encrypted traffic is captured when users visit DApps through their PCs.

Table 1: The scale of DApps encrypted traffic dataset

Categories	DApps (Number of flows)
Exchanges	1inch(2703), SushiSwap(1508), dYdX(1641), Curve(1460), Matcha(6690), Nash(2023), Mirror(2649), Tokenlon(2356)
Development	Enigma(1543), Rocket Pool(3822), Aelf(999), MyContract(1573)
Finance	Tether(1032), MakerDAO(3863), Nexo(1016), AaveP(4513), Paxos(3495), Harvest(4589), Ampleforth(926), Synlev(1041), BarnBridge(3108)
Gambling	Dice2win(1551), FunFair(1062), Edgeless(2059), Kingtiger(1129)
Governance	Kleros(1510), Decentral Games(1536), Iotex(1026), Aragon(3574)
Marketplace	Knownorigin(2009), Ocean Market(6462), OpenSea(2099), Superrare(1306), District0x(1011), Cryptokitties(4722)
Media	AVXChange(46282), Refereum(1074), Megacryptopolis(13924)
Game	Axie Infinity(1049), BFH(3415), Evolution Land(1516), F1deltatime(6119), Nftegg(1006), Pooltogether(2353)
Property	Decentraland(3932), FOAM(6558)
Social	Livepeer(1337), Loom Network(1073), Catallax(1219), 2key(3705)
High risk	DoubleWay(1045), E2X(1054), Gandhiji(1143), Forsage(3501), Minereum(4557)
Security	Chainlink(1173), Quantstamp Security Network(3203)
Storage	Storj(1020), Numerai(2107)
Identity	LikeCoin(1021), SelfKey(1521)

To construct out DApps dataset, we select top-60 DApps on Ethereum with the most users [2]. The categories contain social, gambling, finance, marketplaces, etc. We collect 199,513 DApps traffic flows. The number of flows for each DApps is summarized in Tabel 1. The dataset is split into training and testing sets. And the validation set is constructed by random sampling from the training set (i.e., train : val : test = 7 : 1 : 2)

5.2 Experiments Settings

Baselines To evaluate CQNet, we leverage the following six typical methods for comparison.

FFP [18], which fuses time series, packet length and burst features to the high-dimensional feature through a kernel function, which is fed to Random Forest classifier.

APPS (Appscanner) [21], which captures statistical features of packet length, such as mean, minimum, standard deviation of incoming, outgoing, and bi-directional flows. Then using Random Forest to identify mobile applications.

RF+LT [24], which extracts packet length sequence and time series. Then using Random Forest classifier to identify encrypted flows from 11 DApps.

DF+D (DeepFingerprinting) [20], which only uses the information of packet direction from flows, and sends it into a Convolutional Neural Networks to classify the encrypted flows of websites.

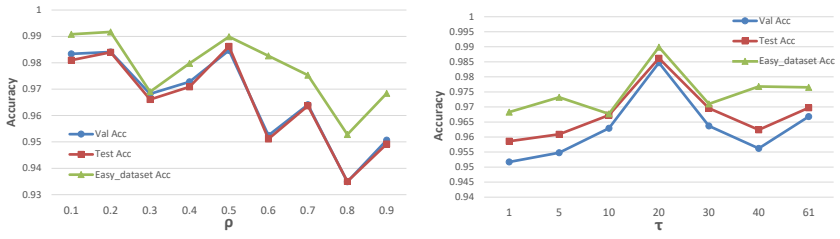
FS-net [10], which takes multi-layer bi-GRU encoder and decoder to learn features of flow sequence, which are utilized for classification of 18 applications.

DF+L, which uses the same Convolutional Neural Networks as DF+D, but the input is the packet length sequence.

Setting of the CQNet We take the raw traffic data as the input of the CQNet. The feature extractor contains four layers. The first two are convolutional (i.e.,32 and 64 kernels of size of 5*5) and the others are fully-connected. As for the quadruplet network, it has four branches with shared weight, one of which has five layers. The first three are convolutional (i.e.,32 kernels of size of 9*9, 64 kernels of size of 5*5, and 128 kernels of 3*3) and we replace the output dimension of the final FC layer to be a common metric space without a softmax layer. The SGD optimizer with learning rate of 0.0005 is used and the batch size is 64.

5.3 Hyperparameters of CQNet

CQNet introduces two hyperparameters ρ and τ to control the easy dataset scale and the selection range of negative samples, respectively. We select 30 epochs



(a) Hyperparameters of FE-set (b) Hyperparameters of network

Fig. 5: Performance of CQNet with respect to different value of ρ and τ.

to compare with other methods. When the epoch is larger than 30, the accuracy increment becomes trivial, less than 0.003, but the training time increases significantly. The setting of 30 epochs is also used in [20].

First, we fix τ to 20 and train CQNet with different values of ρ (i.e., from 0.1 to 0.9, the interval is 0.1) and show the results in Fig. 5(a). The ρ represents the amount of data selected as the easy dataset after clustering. The smaller the value of ρ, the more samples are used to train the quadruplet network, the higher time-cost required. As such, we suggest setting ρ to 0.5.

Second, we fix ρ to 0.5. Different values of τ (i.e., 1, 5, 10, 20, 30, 40, 61) are set and the results are shown in Fig. 5(b). We find that the optimal value is around 20. Therefore, it is recommended to set τ with 20.

Table 2: Accuracy and time-cost comparisons on DApps traffic classification. The metrics of time-cost includes feature extraction time (FET), classifier training time (CTT), data testing time (DTT)

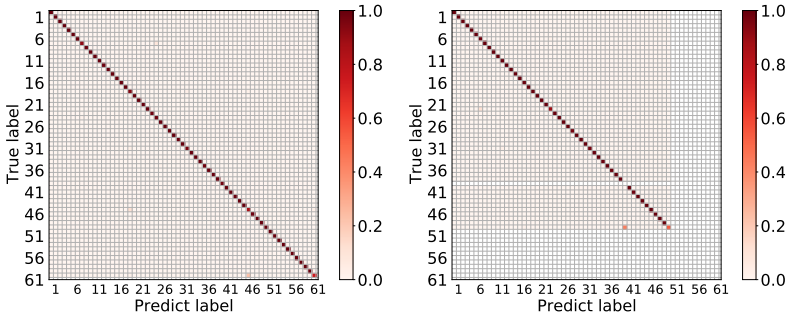
Method	FFP	APPS	RF+LT	DF+D	DF+L	FS-net	CQNet
Accuracy	0.9122	0.8437	0.8977	0.6502	0.7849	0.9611	0.9837
One epoch CTT	-	-	-	122.65	124.73	137.06	94.58
FET	2230.65	1895.19	1589.49	119.91	113.31	12.52	258.21
CTT	352.05	114.52	203.1	3724.96	3668.14	4184.28	2894.59
DTT	8.19	9.54	8.77	10.33	10.28	12.81	32.89
Total time	2590.89	2019.25	1801.36	3855.2	3791.73	4209.61	3185.69

5.4 Performance Comparison

We use **accuracy** and **time-cost** as the performance metrics, which is defined as the proportion of all DApps flows that are classified correctly. And the comparison results are shown in Table 2. We can make the following observations.

First, we can observe that CQNet outperforms the other methods, with the highest accuracy (0.9837 for hard dataset, 0.9897 for easy dataset), indicating that clustering-based metric learning is helpful to the task of DApps encrypted traffic classification. The confusion matrices of CQNet are shown in Fig. 6. Most of DApps can be accurately identified, while the classification accuracy of one DApp is lower than 0.8. To test the easy dataset similarity to all categories, we keep the centers of clusters which belong to NL, so there is no value in a row.

Second, machine learning and deep learning have different emphases on time-cost. The former requires manual extraction of more effective features, while the latter automatically learns more effective features, thus eliminating the need for many experts. The time-cost of our network is the lowest among all compared to deep learning methods. Therefore, it proves that our method can improve the overall efficiency of classification.



(a) Hard dataset (b) Easy dataset

Fig. 6: Confusion Matrices of CQNet on hard dataset and easy datset.

5.5 Ablation Studies

We evaluate the contribution of each of CQNet's components. The baseline is the quadruplet network with fixed loss and uses the original dataset to train and test (denoted as base). The other settings are baseline with distance-based quadruplet loss (denoted as base+d), baseline using the hard dataset (denoted as base+h), baseline with all CQNet's components (i.e., CQNet).

Accuracy	base	base+d	base+h	CQNet
Val Acc	0.9539	0.9691	0.9792	0.9837
Test Acc	0.9551	0.9714	0.9796	0.9842
Easy dataset Acc	-	-	0.9862	0.9897

Table 3: Performances by different components

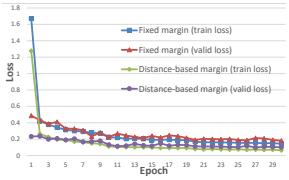
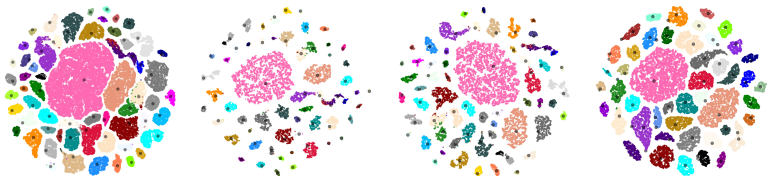


Fig. 7: Comparison results: fixed margin and distance-based margin.

The results are shown in Table. 3. We can see that baseline performs the worst, but it still achieved 95.39% accuracy. Compared with the base model, the improvement of base+d indicates that the distance-based quadruplet loss is beneficial for DApps traffic classification. From Fig. 7, the proposed loss is smoother than the fixed margin loss. Besides, the improvement of base+h is more prominent and the efficiency is also greatly improved because of the reduction of training samples. As expected, CQNet achieves the best results.

To provide a more intuitive understanding of the cluster center, we visualize with four sets as shown in Fig. 8. It can be seen that the samples relevant to each anchor form a cluster around it and the radius of most categories are different, which also confirm the main idea of CQNet. Since not all categories have samples in the easy dataset, there is no sample around the few centers in Fig. 8(d).



(a) Training set (b) Validation set (c) Testing set (d) Easy set

Fig. 8: Visualization to show the performance of our CQNet on different sets by using t-SNE. The center of each category is denoted as a black dot.

6 Conclusion

In this paper, we proposed CQNet, an approach to filter out easy samples based on a clustering algorithm and classify DApps encrypted traffic through metric learning. The easy dataset and hard dataset are split by FE-set algorithm, which includes mini-batch KMeans algorithm, Kuhn-Munkres algorithm, etc. The hard dataset and center set of clusters are sent to the quadruplet network, which can leverage more restrictive relationships among samples. We validate the effectiveness of CQNet on the real-world network traffic, including above 60 DApps. The experimental results show that CQNet outperforms the state-of-the-art methods. Besides, using the trained classifier to test the easy dataset and its accuracy is higher than the testing set in the hard dataset.

Acknowledgements This work is supported by The National Key Research and Development Program of China (No.2020YFB1006100, No.2020YFE0200500 and No.2018YFB1800200) and Key research and Development Program for Guangdong Province under grant No. 2019B010137003.

References

1. Ethereum, `https://www.ethereum.org/`. Last accessed 21 June 2021
2. DApps store 2021, `https://www.stateofthedapps.com/`. Last accessed 21 June 2021
3. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: website fingerprinting attacks and defenses. In: the ACM Conference on Computer and Communications Security, CCS. pp. 605–616 (2012)
4. Conti, M., Mancini, L.V., Spolaor, R., Verde, N.V.: Analyzing android encrypted network traffic to identify user actions. IEEE Trans. Information Forensics and Security 11(1), 114–125 (2016)
5. Feghhi, S., Leith, D.J.: A web traffic analysis attack using only timing information. IEEE Trans. Inf. Forensics Secur. 11(8), 1747–1759 (2016)
6. Grolman, E., Finkelshtein, A., Puzis, R., Shabtai, A., Celniker, G., Katzir, Z., Rosenfeld, L.: Transfer learning for user action identication in mobile apps via encrypted trafc analysis. IEEE Intell. Syst. 33(2), 40–53 (2018)

7. Hayes, J., Danezis, G.: k-fingerprinting: A robust scalable website fingerprinting technique. In: 25th USENIX Security Symposium. pp. 1187–1203 (2016)
8. Korczynski, M., Duda, A.: Markov chain fingerprinting to classify encrypted traffic. In: INFOCOM. pp. 781–789 (2014)
9. Li, R., Xiao, X., Ni, S., Zheng, H.: Byte segment neural network for network traffic classification. In: International Symposium on Quality of Service. pp. 1–10 (2018)
10. Liu, C., He, L., Xiong, G., Cao, Z., Li, Z.: Fs-net: A flow sequence network for encrypted traffic classification. In: INFOCOM. pp. 1171–1179 (2019)
11. Liu, H., Wang, Z., Wang, Y.: Semi-supervised encrypted traffic classification using composite features set. J. Networks 7(8), 1195–1200 (2012)
12. Liu, J., Fu, Y., Ming, J., Ren, Y., Sun, L., Xiong, H.: Effective and real-time inapp activity analysis in encrypted internet traffic streams. In: International Conference on Knowledge Discovery and Data Mining. pp. 335–344 (2017)
13. Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., Wehrle, K.: Website fingerprinting at internet scale. In: 23rd Annual Network and Distributed System Security Symposium, NDSS (2016)
14. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: An overview. IEEE Commun. Mag. 57(5), 76–81 (2019)
15. Rimmer, V., Preuveneers, D., Juarez, M., van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. In: 25th Annual Network and Distributed System Security Symposium, NDSS (2018)
16. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. pp. 815–823 (2015)
17. Shen, M., Wei, M., Zhu, L., Wang, M.: Classification of encrypted traffic with second-order markov chains and application attribute bigrams. IEEE Trans. Inf. Forensics Secur. 12(8), 1830–1843 (2017)
18. Shen, M., Zhang, J., Zhu, L., Xu, K., Du, X., Liu, Y.: Encrypted traffic classification of decentralized applications on ethereum using feature fusion. In: Proceedings of the International Symposium on Quality of Service, IWQoS. pp. 18:1–18:10 (2019)
19. Shi, H., Li, H., Zhang, D., Cheng, C., Cao, X.: An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. Comput. Networks 132, 81–98 (2018)
20. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In: CCS. pp. 1928–1943 (2018)
21. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In: IEEE European Symposium on Security and Privacy, EuroS&P. pp. 439–454 (2016)
22. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app identification via encrypted network traffic analysis. IEEE Transactions on Information Forensics and Security 13(1), 63–78 (2017)
23. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking, ICOIN. pp. 712–717 (2017)
24. Wang, Y., Li, Z., Gou, G., Xiong, G., Wang, C., Li, Z.: Identifying dapps and user behaviors on ethereum via encrypted traffic. In: 16th EAI Security and Privacy in Communication Networks, SecureComm. pp. 62–83 (2020)
25. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: Computer Vision - ECCV. pp. 499–515 (2016)
26. Zhou, X., Demetriou, S., He, D., Naveed, M.: Identity, location, disease and more: inferring your secrets from android public resources. In: CCS. pp. 1017–1028 (2013)