GAP-WF: Graph Attention Pooling Network for Fine-grained SSL/TLS Website Fingerprinting

Jie Lu*†, Gaopeng Gou*†, Majing Su‡, Dong Song‡, Chang Liu*†, Chen Yang*†, Yangyang Guan*†✉
*Institute of Information Engineering, Chinese Academy of Sciences
†School of Cyber Security, University of Chinese Academy of Sciences
‡6th Research Institute of China Electronic Information Industry Group Co., Ltd.
Beijing, China
{lujie,gougaopeng}@iie.ac.cn, {jingmasu,songdong248}@163.com, {liuchang,yangchen,guanyangyang}@iie.ac.cn

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533543

***Abstract*—As an important part of network management, website fingerprinting has become one of the hottest topics in the field of encrypted traffic classification. Website fingerprinting aims to identify the specific webpages in encrypted traffic by observing patterns of traffic traces. Prior studies proposed several machine-learning-based methods using statistical features and deep-learning-based methods using packet length sequences. However, these works mainly focus on the website homepage fingerprinting. In fact, people are usually not limited to visiting the homepage. Compared with the homepage classification of websites, it is more difficult to identify different webpages within the same website due to the traffic traces are very similar. In this paper, we propose the Graph Attention Pooling Network for fine-grained website fingerprinting (GAP-WF). We introduce the trace graph to describe the contextual relationship between flows in webpage loading. Then we utilize the Graph Neural Networks to learn the intra-flow and inter-flow features. Considering different flows may have different importance, we utilize the graph attention mechanism to pay attention to key nodes. We collect four datasets covering three different granularity scenarios to evaluate our proposed method. Experimental results demonstrate that GAP-WF not only achieves the best performance of 99.86% in website homepage fingerprinting, but also outperforms other state-of-art methods in all fine-grained webpage fingerprinting scenarios. Moreover, GAP-WF can achieve better performance with fewer training samples.**

***Index Terms*—Website Fingerprinting, Encrypted Traffic Classification, SSL/TLS, Graph Attention Networks, Deep Learning**

I. Introduction

With the gradual improvement of people's network security awareness, Encryption protocols such as SSL/TLS [1], [2] are widely used in various websites to ensure security. According to W3Techs [3], as of January 2021, default protocol HTTPS is used by 69.2% of all the websites. However, while SSL/TLS protects privacy, it also brings challenges to network management (e.g. QoS and malicious behavior tracking). Recently, as an important part of network management, *Website Fingerprinting (WF)*, which aims to identify the specific webpages from encrypted traffic, has become one of the hottest topics in cyberspace security [4]–[15].

WF mainly studies how to extract effective features from *traffic traces* [5], [14] to achieve better classification performance. The traffic trace includes several flows generated by webpage loading, and the flows represent different resources requested (e.g. HTML, JavaScript, and third-party libraries). Previous studies [6]–[9] proposed several machine learning (ML) based methods using statistical features. These works extract hand-crafted features from traffic traces and then feed them to classifiers such as Support Vector Machine (SVM) and Random Forest (RF). Recently, with the development of deep learning (DL) techniques, several DL-based methods [13]–[15] have been proposed. These methods use raw packet length sequence as input and utilize models such as Convolutional Neural Networks (CNN) to learn the high-level features. Since complex hand-crafted feature selection is no longer required, the DL methods gradually become the mainstream.

However, these studies simply regard WF as a website homepage classification problem. In fact, people are usually not limited to visiting the homepage. The homepage traffic can not fully represent a website [10]. Different webpages within a website may represent different services, which should be treated differently in QoS and behavior tracking. Different from website homepage fingerprinting, the layout and content size of different webpages within a single website are very similar, the features of traffic traces proposed by previous works may be no longer distinguishable [8]. As a result, the performance of the previous methods is significantly reduced. Although recent studies [4], [5] propose to combine global coarse-grained features with local fine-grained features to improve the performance of fine-grained WF, these methods are based on ML and need complex hand-crafted feature selection.

Recently, several traffic analysis studies [16], [17] have proven that the graph structure can well represent the global and local information (e.g. the contextual relationship between flows) of traffic traces. Meanwhile, the Graph Neural Networks (GNN) [18], which has been widely used in other fields, can effectively capture hidden patterns of graph structure data. Therefore, we can utilize the GNN to learn the effective features of fine-grained webpage traffic traces.

In this paper, we propose Graph Attention Pooling Network for fine-grained WF (GAP-WF). We construct the *trace graph* according to the flow sequence in webpage loading and use a GNN-based model to better learn the features of nodes (intra-flow) and structure (inter-flow). Moreover, considering

Corresponding author: guanyangyang@iie.ac.cn

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

different flows may have different effects on classification (e.g. the HTML request flows may be the key nodes in the graph, and the flows of third-party libraries may even have a negative impact and become noisy nodes), the reasonable model needs to treat these flows differently. Therefore, we utilize *Graph Attention Network (GAT)* to pay attention to the more useful nodes. Besides, we build four datasets covering three types of WF scenarios to evaluate the performance of GAP-WF. The experimental results show that GAP-WF outperforms other state-of-the-art methods.

Our contributions can be briefly summarized as follows:

1) We propose a novel method named GAP-WF for fine-grained website fingerprinting. The GAP-WF introduces the trace graph to describe the contextual relationship between flows in webpage loading. Moreover, GAP-WF utilizes a GNN-based model to better learn both the intra-flow and inter-flow features.
2) We utilize the multi-head graph attention mechanism to boost the performance of WF. By learning the importance of each flow, the potential key nodes can be acquired, which can reduce the negative impact of noisy nodes and improve the classification performance.
3) We collect four datasets covering three different granularity scenarios: traditional website homepage classification scenario, webpage classification scenario of a single website, and webpage classification scenario of multiple websites. Experiment results show that the proposed model achieves the best performance on all datasets and can improve the precision by 15.7% than the state-of-the-art methods. Moreover, our GAP-WF can achieve better accuracy with fewer training samples.

The remaining paper is organized as follows. Section II reviews the related works. Section III describes the proposed model in detail. The comparison experiments are conducted in Section IV. Finally, we conclude this paper in Section V.

II. Related Work

In this section, we first review several state-of-the-art works of WF, including ML-based methods and DL-based methods. Then, we introduce some graph-based methods for traditional traffic analysis tasks.

A. Website Fingerprinting Methods

As early as 2005, Bissias et al. [19] first proposed using IP packet size and inter-packet arrival times to implement WF. Since then, series of WF methods based on ML or DL have been proposed and are used to benchmark other methods.

1) Machine-learning-based methods: Wang et al. [9] proposed a WF method named k-NN. This method used statistical features such as the number of incoming and outgoing packets and the number of bursts. The authors used a K-Nearest Neighbors (kNN) classifier to measure the similarity between the features of different websites.

Panchenko et al. [8] proposed a WF model called CUMUL based on a Support Vector Machine (SVM) classifier. The method samples fixed-size features from the cumulative sum of packet sizes to represent traffic traces. They not only identified the homepage of the websites but also studied fine-grained webpage classification scenarios. They divide the fine-grained datasets into two cases: a website is only represented by the homepage, and a website is given by a subset of its webpages.

Hayes et al. [7] proposed the k-fingerprinting (KFP). KFP extracts 150-dimensional statistical features from the traffic traces, then uses a Random Forest classifier (RF) to extract fingerprints of websites. They analyze the importance of features and found that the total number of incoming packets is the most informative feature. Besides, in order to solve the open-world problem, the authors use the leaves of the trees in RF as the feature vectors of the sample, then feed these vectors to a k-Nearest Neighbor (k-NN) classifier to get the final decision.

Recently, to solve the problem of fine-grained WF, especially in HTTP/2 scenarios, Zhang et al. [4] propose local request and response sequence feature (LRRS), which include the global and local features of webpage traffic. They utilized the Deep Forest [20] algorithm to learn the sequential relationships in features. This algorithm can learn features better than traditional machine learning methods. They studied three different granularity scenarios, including the traditional homepage dataset, the webpage dataset in a single website, and the webpage dataset in multiple websites.

Shen et al. [5] propose a fine-grained webpage fingerprinting method called FineWP using the statistical features of packet length information. This method uses the RF as the classifier. They only study a scenario of webpage classification in a single website.

2) Deep-learning-based methods: Rimmer et al. [15] proposed to apply three DL models to WF: Stacked Denoising Autoencoder (SDAE), Convolutional Neural Network (CNN), and Long-Short Term Memory (LSTM). These deep learning models are capable of automatically learning traffic features for WF. The CNN-based model, also known as *Automated Website Fingerprinting (AWF)*, achieves the best performance between the three models. However, this method could not outperform state-of-the-art WF methods such as CUMUL.

Sirinam et al. [14] proposed Deep Fingerprinting (DF), which use a CNN model with a sophisticated architecture design. DF only takes the traffic direction of each packet as input features in the form of $[-1, +1]$. They only studied the website homepage fingerprinting, and this method has become the current state-of-the-art WF method.

Recently, to solve the problem of the high cost of model retraining caused by the change of data distribution, Sirinam et al. [12] proposed Triplet Fingerprinting (TF). It utilizes N-shot learning (NSL) to achieve high accuracy using fewer examples per website. This method has two phases: pre-training phase and attack phase. The pre-training phase uses the DF as the feature extractor.

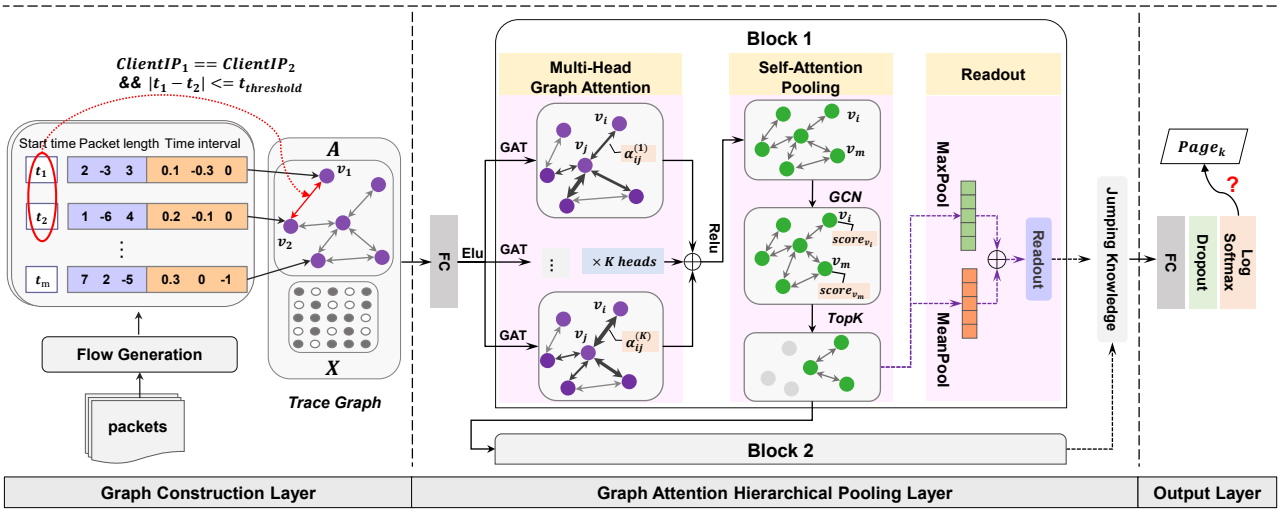


Fig. 1. The system overview of the GAP-WF.

B. Graph based traffic analysis methods

Although GNN has not been used to implement WF before, several recent works proves that traffic graph is helpful for traffic analysis [16], [17], [21]. To classify mobile app traffic, FlowPrint [16] constructs the correlation graph according to the temporal correlation between flow clusters. The cluster represents a set of flows with the same destination IP or TLS certificate. Then the IP and certificates are used as features, and the Jaccard similarity was used to match fingerprints. However, this method relies on the plain text segments. It is not applicable for fine-grained WF due to webpages within the same website usually have the same SNI and certificate.

In order to use few labeled data to achieve higher accuracy in traffic classification, GCN-TC [17] constructs the traffic trace graph in which a node represents a flow and edges denote two nodes have common IP. Then GCN is used to learn effective abstract features. However, the method mainly focuses on application layer protocol classification and adopt statistical features as input.

III. The Proposed Method

Our GAP-WF model consists of 3 parts as shown in Fig. 1. The *Graph Construction Layer* constructs trace graphs from the pcap files. Then the *Graph Attention Hierarchical Pooling Layer* is used to extract the graph embedding. And the last part is *Output Layer*. In this section, we will present each part in detail.

A. Graph Construction Layer

As we mentioned before, the target of WF is to classify specific webpage traffic traces, which refers to a collection of packets of the same client IP in a period of time. The traffic traces can be split into several flows through five-tuples. In our proposed model, as shown in Fig. 1, we first split the packets into flows through *Flow Generation* module. And we record the packet length sequence, time interval sequence, direction sequence, and the first packet (generally is *SYN*) timestamp t_i of each flow. Then we use these four types of information to construct traffic trace graphs. Where a graph $\mathcal{G}$ represents a trace, a node i represents a flow.

1) Graph Representation: Generally, a graph is represented as $\mathcal{G} = (V, E)$, where V is the set of nodes, and E is the set of edges [22]. Assuming the graph has N nodes, and each node has D-dimensional features. Then the adjacency matrix $A \in \mathbb{R}^{N \times N}$ and the feature matrix $X \in \mathbb{R}^{N \times D}$ are taken as the inputs of graph attention hierarchical pooling layer.

2) Node Feature Representation: In WF, three types features are generally used: $packet\_size$, $time\_inter$ and $direction$. Prior work has shown that the most important features are derived from the lengths of traces in each direction [9], [14]. Recent methods (e.g. DF) do not use temporal features. However, we performed preliminary evaluations to compare the WF performance between using $time\_inter$ and without $time\_inter$. The result shows that in the homepage WF task, the accuracy is already over 99% without temporal features. But in the fine-grained webpage tasks, temporal features improved the accuracy from 88% to 93%. Therefore, in homepage WF scenario, we only use $\pm packet\_size$ to represents the feature $\vec{x}_i$ of node i. And in webpage scenario, we use $(\pm time\_inter, \pm packet\_size)$. Where the sign of $packet\_size$ and $time\_inter$ indicates the direction of the packet: positive means outgoing and, negative, incoming.

3) Edge Representation: In our graph, node i and node j have edge e_{ij} if and only if the two nodes satisfy all the following rules:

$$ClientIP_i == ClientIP_j \tag{1}$$

$$|t_i - t_j| <= t_{threshold} \tag{2}$$

where the *ClentIP* denotes the identity of the client, which can be easily distinguished by port number, because usually the server port of TLS/SSL traffic is 443. The $t_{threshold}$ is a prior knowledge, denotes the average webpage load time. According

to Contentsquare [23], the average webpage load time of all industries in 2020 is 2.39 seconds, and the average time spent per webpage is 62 seconds. This means that when the user clicks a link in the browser, all the resources requested are loaded in first 2.39 seconds. Therefore, we set the $t_{threshold}$ to 2.39, and the result show that such a value can achieve sufficiently high accuracy.

In this way, we construct an adjacency matrix $A \in \mathbb{R}^{N\times N}$ and a feature matrix $X \in \mathbb{R}^{N\times D}$ for each traffic trace. Finally, we use PyTorch Geometric Library (PYG) [24], which is one of the most popular GNN libraries, to create the graph dataset.

B. Graph Attention Hierarchical Pooling Layer

As shown in Fig. 1, the *Graph Attention Hierarchical Pooling (GAP) Layer* mainly consists of 2 convolution blocks, and each block consists of 3 sublayers: *Multi-Head Graph Attention (MGAT)* Layer, *Self-Attention Graph Pooling (SAGPool)* Layer and *Readout* Layer.

First, in order to obtain sufficient expressive power, we normalize the input and transform the features into higher-level features X' through a full connection network. We use the *Exponential Linear Unit (ELU)* activation function to handle negative inputs [14]. Then the adjacency matrix A and new feature matrix X' are fed into the MGAT layer.

1) Multi-Head Graph Attention Layer: The MGAT layer aims to extract effective graph embedding from graph structure data. In order to utilize both the structure and node information of graph data, *Graph Neural Networks (GNN)* models such as *Graph Convolution Networks (GCN)* are proposed. GCN can perform convolution operations on graph data [18]. The main idea is to update the hidden representation of nodes by passing and aggregating the features of their neighbor nodes. A single-layer GCN can process the information on first-order neighbors. Similar to CNN, GCN stack multiple graph convolutional layers to extract high-level node representations. The layer-wise propagation rule is as follows:

$$H^{(l+1)} = \sigma(\check{D}^{-\frac{1}{2}}\check{A}\check{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{3}$$

where I is the identity matrix, $\check{A} = A + I$ is the adjacency matrix of the undirected graph with added self-connections. $\check{D}$ is the degree matrix of $\check{A}$, W^l is a layer-specific trainable weight matrix, $H^{(l)}$ is the feature matrix in the l^{th} layer; $H^{(0)} = X$, and $\sigma(\cdot)$ denotes an activation function, such as ReLU. GCN plays a central role in building up many other complex GNN models [22]. However, it treats the features of each node equally when it updates the hidden representation of the nodes.

As we mentioned before, a traffic trace contains several flows, which correspond to different resources requested (e.g. HTML and third-party libraries). Different flows may have different importance to classification, the HTML request flows may be the key nodes in the graph, and the flows of third-party libraries may even have a negative impact and become noisy nodes. The reasonable model should pay more attention to these key nodes while reducing the negative impact of noisy nodes. Therefore, we utilize the *Graph Attention Network (GAT)* [25], which introduces the attention mechanisms into GCN. It can automatically learn the weight of neighbor nodes by calculating the similarity between nodes. The attention coefficients, which indicate the importance of node j's features to node i, are expressed as:

$$\alpha_{ij} = \frac{exp(LeakyReLU(\vec{a}^T[W\vec{x}_i \parallel W\vec{x}_j]))}{\sum_{k\in\mathcal{N}_i} exp(LeakyReLU(\vec{a}^T[W\vec{x}_i \parallel W\vec{x}_k]))} \tag{4}$$

where $\cdot^T$ represents transposition and $\parallel$ is the concatenation operation. $\mathcal{N}_i$ is the neighbors of node i in the graph, $\vec{x}_i$ is the feature of node i. W is a weight matrix that is applied to every node, and $\vec{a}$ is a weight vector of a single-layer feedforward neural network. The W and $\vec{a}$ are learnable parameters. The LeakyReLU function is used to enhance nonlinear expression ability. Then the softmax function is used to calculate the final attention coefficients.

Then the normalized attention coefficients are used to compute the final features for every node. Since the GAP-WF utilize the *multi-head attention* to improve the expression ability of the attention layer, the new feature vector of node i is expressed as follows:

$$\vec{x}'_i = \parallel_{k=1}^{K} \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(k)} W^{(k)}\vec{x}_j) \tag{5}$$

where K is the number of heads, $\alpha_{ij}^{(k)}$ are normalized attention coefficients computed by the k-th attention mechanism.

2) Self-Attention Graph Pooling Layer: Graph Pooling enable GNN models to reduce the number of parameters by scaling down the size of representations, and thus avoid overfitting. In our model, we utilize a hierarchical pooling method called SAGPool [26]. It can consider both node features and graph structure due to GCN is used to calculate the self-attention score of nodes. The score $S \in \mathbb{R}^{N\times 1}$ is calculated as follows.

$$S = \sigma(\check{D}^{-\frac{1}{2}}\check{A}\check{D}^{-\frac{1}{2}}X\Theta_{att}) \tag{6}$$

where $\Theta_{att} \in \mathbb{R}^{D\times 1}$ is the only parameter of SAGPool. Once all the nodes obtain a self-attention score, the SAGPool adopts the topK mechanism, which only retains the nodes with the top kN scores.

$$idx = toprank(S, \lceil kN \rceil), S_{mask} = S_{idx} \tag{7}$$

$$X' = X_{idx,:}, X_{out} = X' \odot S_{mask}, A_{out} = A_{idx,idx} \tag{8}$$

where $k \in (0, 1]$ is pooling ratio, toprank returns the indices of the top $\lceil kN \rceil$ scores, $\cdot_{idx}$ is an indexing operation, S_{mask} is the feature attention mask. $\odot$ is the broadcasted element-wise product, $X_{idx,:}$ is the node-wise indexed feature matrix, X_{out} is the new feature matrix and A_{out} is the corresponding adjacency matrix.

Then the pooled graph is fed to two parts, one is the next block, and the other is the *readout layer*.

TABLE I
THE DETAILED INFORMATION OF FOUR DATASETS

Dataset Name	Description	Class Number	Instances in Each Class	Number of flows
WEB100	100 websites and each website with only homepage	100	100	329997
APPLE60	60 popular product pages on apple.com	60	100	35437
CDC30	30 most searched disease-related webpages on cdc.gov	30	100	29907
PAGE100	100 pages of 9 websites	100	100	238620

3) Readout Layer: To solve the graph classification problem, we need to extract the graph global representation by *readout* operation. The readout can aggregate node features to make a fixed-size representation. Similar to CNN, readout supports some operations such as Sum, Mean, and Max. In our model, we adopt *Maxpool* and *Meanpool*. The output of the readout is as follows.

$$\vec{r} = \frac{1}{N}\sum_{i=1}^{N}\vec{x}_i \parallel \max_{i=1}^{N}\vec{x}_i \tag{9}$$

Similar to the skip connection in CNN, *Jumping Knowledge (JK)* aggregates the graph readout of each convolution block into a global final representation [27]. JK has three types of aggregation operations: concatenation, max, and LSTM-attention. LSTM-attention is the most complex approach, which learns an attention score for each readout. However, our result shows that the simple concatenation operation can obtain the best performance.

C. Output Layer

The output layer has 3 parts. First, a full connection operation is used to convert the feature dimension to the output size. Then a dropout is adopted to avoid overfitting. Finally, we utilize *Log Softmax* to calculate the probability of each class, which improves efficiency by replacing Softmax's exponential operations with addition and subtraction operations. Correspondingly, we use *NLLLoss* to estimate the loss value.

IV. EVALUATION

In this section, we first introduce our datasets and experiment setting. After that, we compare our model with four state-of-the-art methods. The comparison results are shown in Table III. We utilize *precision*, *recall*, and *f1* as metrics. We also show the precision-recall curves. Moreover, we compare the effect of training size on accuracy in each method. Finally, we analyze whether the graph attention mechanism could improve the performance of WF.

A. Dataset

Since there is no publicly-available webpage traffic dataset, we collect real-world webpage traffic datasets using two PCs, one is equipped with Windows10 and the other is equipped with Ubuntu20.04. Firefox and Chrome are installed on both machines, and Selenium[1] is used to simulate the behavior of humans visiting webpages. We use *tcpdump* and *windump* to capture network traffic and store it as *pcap* format file. We empty the caches each time visiting a certain webpage and close the browser when each visit is completed.

We select three types of fine-grained WF scenarios, including traditional homepage classification scenario, webpage classification scenario of a single website, and webpage classification scenario of multiple websites. Each webpage represents a class, and each class has 100 instances. The four datasets we used in our study are presented in TABLE I.

- **WEB100**. In the homepage classification scenario, we collected a dataset called WEB100. It covers the homepages of 100 websites, which are encrypted with TLS/SSL and randomly selected from the top Alexa 300 websites.
- **APPLE60** and **CDC30**. In the scenario of webpage classification of a single website, we collected two datasets: dataset APPLE60 covers 60 product webpages of www.apple.com, such as www.apple.com/iphone. And dataset CDC30 covers the 30 most searched disease-related webpages on the website www.cdc.gov, such as www.cdc.gov/coronavirus/2019-ncov. These two datasets have different webpage styles. The main content of CDC30 is a lot of text with a few pictures, while APPLE60 mainly contains many pictures and videos.
- **PAGE100**. In the scenario of webpage classification of multiple websites, we selected a total of 100 webpages from 9 websites and denoted the dataset as PAGE100, including entertainment (e.g. bilibili.com), shopping (e.g. ebay.com), health (e.g. cdc.gov), and other types of websites. These webpages are mainly the tabs of the website, representing a certain type of service of the website. For example, www.bilibili.com/v/game provides game services. Usually, these webpages are the most representative webpages on the website.

To build the input, we use Joy[2] to extract the pcap file into a JSON that containing the relevant features. Note that we discard packets without payload (e.g. ACK) or retransmission, which follows the previous work [11], [14]. The data collection is conducted within a period of a month.

B. Experiment Setting

1) Methods to Compare: To have a more comprehensive comparison of our model, we select four state-of-the-art models: CUMUL proposed by Panchenko et al. [8], KFP proposed by Hayes et al. [7], DeepForest proposed by Zhang et al. [7] and DF proposed by Sirinam et al. [14]. Note that we apply RF

[1]https://www.selenium.dev/

[2]https://github.com/cisco/joy

TABLE II
CLASSIFICATION PRECISION %, RECALL % AND F1 SCORE % OF THE PROPOSED MODEL WITH THE COMPARISON OF FOUR STATE-OF-THE-ART METHODS ON FOUR DATASETS.

Model	WEB100			APPLE60			CDC30			PAGE100		
	pre	recall	f1	pre	recall	f1	pre	recall	f1	pre	recall	f1
GAP-WF	**99.86**	**99.85**	**99.85**	**97.10**	**96.74**	**96.72**	**91.62**	**91.50**	**91.45**	**93.77**	**93.40**	**93.37**
DeepForest	98.90	98.80	98.85	93.89	93.41	93.65	87.76	87.16	87.46	76.10	75.70	75.90
DF	99.11	99.04	99.05	83.65	82.91	82.85	70.14	70.00	68.60	75.12	74.40	74.10
KFP	97.68	97.60	97.58	86.81	86.25	86.24	76.65	76.33	75.73	78.08	77.85	77.36
CUMUL	94.10	93.80	93.75	88.15	87.42	87.47	76.55	76.33	76.04	71.34	70.95	70.60

as the classifier of CUMUL instead of SVM, because CUMUL using RF performs better on our datasets.

The GAP-WF is implemented with Pytorch Geometric, which is an extension library for PyTorch. And we implement the DF with Keras, which using Tensorflow as backend. We practice deep learning experiments on a platform that has three Nvidia Tesla V100 GPUs with 32 GB GPU memory, which can make high parallelism performance of CNNs and GNNs due to cuDNN primitives.

2) Setting of GAP-WF: We take the JSON file, which contains features of the flows in a traffic trace such as packet length sequences with direction and time interval sequences, as the input to our GAP-WF. As we mentioned before, in the homepage scenario, we take $\pm packet\_size$ as the feature of nodes. However, in webpage scenarios, we use $(\pm time\_inter, \pm packet\_size)$. Where the sign of $packet\_size$ and $time\_inter$ indicates the direction of the packet: positive means outgoing and negative means incoming.

We conduct an extensive search through hyperparameter space to make the classifier achieve its best performance. Table III shows the selected hyperparameters. It is worth mentioning that we performed statistical analysis on the flow length. As shown in Fig. 2, the length of most flows is less than 60. And we find that only 30 dimensions are needed to achieve high accuracy of 99.85% in the homepage scenario. However, in the fine-grained webpage scenario, 200 dimensions are needed to achieve optimal accuracy due to more information is needed to find out the potential differences between the webpages under the same website.

TABLE III
HYPERPARAMETERS SELECTION FOR GAP-WF

Hyperparameters	Search Range	WEB100	APPLE60 CDC30 PAGE100
Input Dimension	[10...2500]	30	200
Hidden Dimension	[128...1024]	512	512
Optimizer	[Adam, SGD]	Adam	Adam
Learning Rate	[0.0001...0.01]	0.001	0.001
Dropout Rate	[0.1...0.6]	0.1	0.1
Pooling Ratio	[0.1...0.9]	0.8	0.8
Weight Decay	[0.0001...0.01]	0.0001	0.0001
Batch Size	[16...512]	64	64

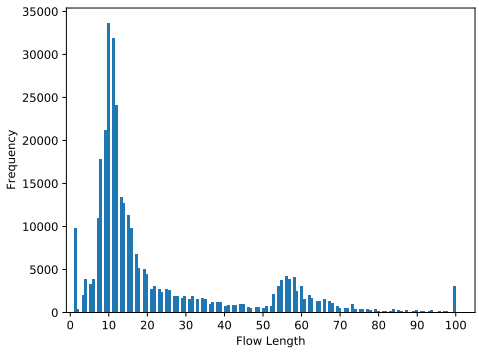


Fig. 2. The frequency histogram of flow length of dataset WEB100, where the flow length refers to the number of packets whose payload is not empty.

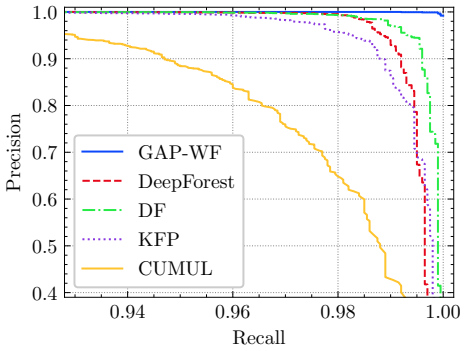


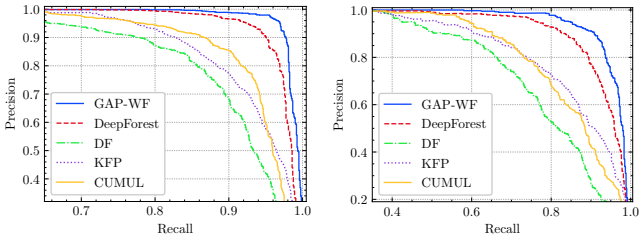
Fig. 3. Precision-Recall curves on WEB100.

C. Website Homepage Classification

As shown in Fig. 3, on WEB100, since the difference in traffic traces between websites is very obvious, all the methods achieve high performance. Especially, GAP-WF attains an f1 score of 99.85%, which is almost the upper limit. It is 0.8% higher than the current state-of-the-art method DF with CNN and 1% higher than the best-performing machine learning method DeepForest. Besides, we find the precision of CUMUL is significantly lower than other methods. Different from other ML-based methods, the CUMUL does not use temporal features. It indicates that the temporal features are useful for ML-based methods in the homepage scenario.

D. Webpage Classification of Single Website

In the fine-grained webpage classification scenario, the traffic traces of different classes become similar to each other. We find that the gap of f1 scores between GAP-WF and other



(a) APPLE60 (b) CDC30

Fig. 4. Precision-Recall curves on APLLE60 and CDC30.

methods increases gradually. The f1 scores of GAP-WF on APPLE60 and CDC30 are 96.72% and 91.45%, respectively, which are 3% to 23% higher than other methods. From Fig. 4, we can observe that among the four comparison methods, only DeepForest, which focuses on fine-grained webpage task, achieves the performance close to GAP-WF. However, the performance of other methods reduces significantly. Besides, we observe that CUMUL performs relatively well on APPLE60 than WEB100. DF, the second place on WEB100, performs the worst on APPLE60 and CDC30. It indicates that only the coarse-grained packet length sequence features cannot distinguish fine-grained webpages well.

Moreover, by comparing the results of these two datasets, we can find that the distinguishability of fine-grained webpages is related to the website. The CDC30 is slightly more difficult than APPLE60. However, our model maintains a large precision gap with other methods. It indicates that our model is stable on different websites.

E. Webpage Classification of Multiple Website

In this scenario, models not only need to pay attention to the differences between multiple websites, but also need to find the subtle differences of similar webpages under the same website. Fig. 5 shows that GAP-WF significantly outperforms other methods on PAGE100. The GAP-WF obtains a precision of 93.77% and an f1 score of 93.37%. However, none of the four state-of-the-art methods achieve an f1 score of 80%.

Overall, as the task becomes more difficult, the gap between GAP-WF and other methods gradually increases. GAP-WF achieves high accuracy on all datasets, while other methods only perform well on part of these datasets. It suggests that GAP-WF is the most robust among comparison methods.

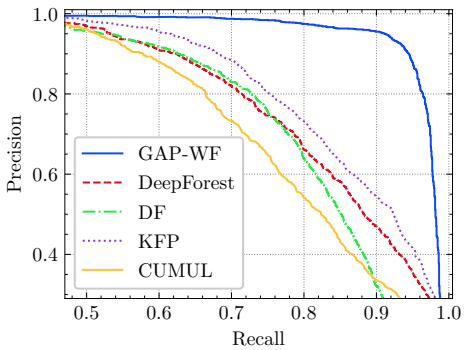
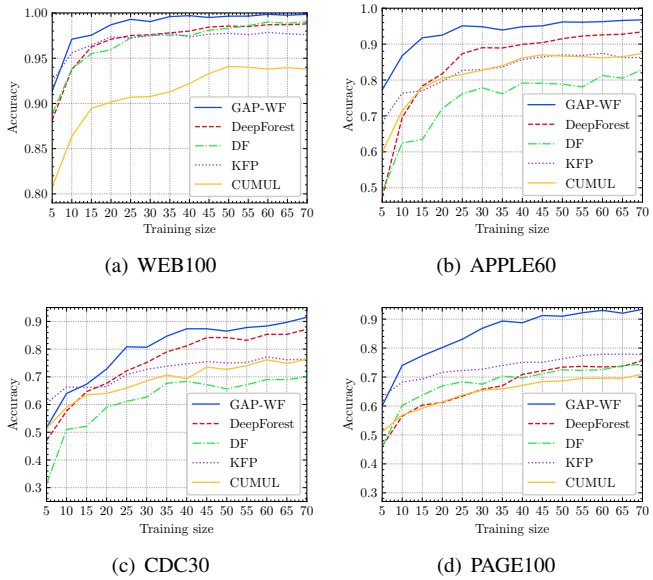


Fig. 5. Precision-Recall curves on PAGE100.



(a) WEB100 (b) APPLE60

(c) CDC30 (d) PAGE100

Fig. 6. Impact of numbers of training traces on classification accuracy

From the results, it is clear that the proposed method can achieve higher classification performance by learning the effective fine-grained representation from intra-flow and inter-flow features.

F. Impact of Training Size

As shown in Fig. 6, our GAP-WF can achieve higher accuracy with fewer training samples. On WEB100, with just 25 instances, GAP-WF achieves 99% accuracy. DeepForest and DF require 60 samples to reach the same accuracy. On APPLE60, GAP-WF only needs 15 instances to achieve an accuracy of 90%, which is higher than the best performance of CUMUL, KFP, and DF.

From Fig. 6(c), we find that the accuracy of DeepForest and GAP-WF have similar growth trends on CDC30. Starting from 25 training samples, the growth rate of DF, KFP, and CUMUL slow down significantly. On PAGE100, we observe that the gap between GAP-WF and other methods increases significantly after 10 samples. Moreover, unlike the first two scenarios, only GAP-WF maintains a high growth rate.

In fact, since webpages are constantly updated, which can reduce the performance of the old models [10], [12]. So a reasonable model should be able to update the model with fewer new samples. Our results show that, compared with other models, our GAP-WF better meets this requirement.

G. Impact of Graph Attention mechanism

In order to prove the graph attention networks can improve accuracy, we conduct an ablation experiment: GCN-WF. Note that the GCN-WF only replaces the multi-head attention layer of GAP-WF with GCN.

As shown in Table IV, the accuracy of GAP-WF is only 0.05% higher than GCN-WF on WEB100.

Since the task of the homepage WF is relatively easy, the accuracy of GAP-WF and GCN-WF is almost the upper limit,

TABLE IV
CLASSIFICATION ACCURACY % OF GAP-WF AND GCN-WF ON FOUR DATASETS.

Model	Accuracy			
	WEB100	APPLE60	CDC30	PAGE100
GAP-WF	99.85	96.75	91.50	93.40
GCN-WF	99.80	95.80	89.17	90.15

so the improvement of the attention mechanism is not obvious. However, on APPLE60 and CDC30, the gap between GAP-WF and GCN-WF increases to more than 1%. On PAGE100, the gap between GAP-WF and GCN-WF increases to 3.3%. Overall, as the similarity between classes increases, the advantages of the multi-head attention layer is more obvious. The results demonstrate that the graph attention mechanism is helpful to improve the website fingerprinting accuracy.

V. CONCLUSION

In this paper, we proposed a novel fine-grained website fingerprinting method named GAP-WF. To effectively represent the patterns of webpage loading, GAP-WF constructs the trace graph according to the resource requested flow sequence. Moreover, to better learn the intra-flow and inter-flow features, GAP-WF utilizes multi-head GAT to pay attention to the potential key flows while reducing the negative impact of noisy flows. We collect four datasets in three different granularity scenarios to evaluate the performance of the proposed model. Experimental results show that GAP-WF not only achieves the best performance of 99.86% on the traditional homepage WF dataset but also outperforms other state-of-art methods on all fine-grained WF datasets. Especially on the PAGE100 dataset, the precision of GAP-WF is 15.7% to 22.4% higher than the state-of-the-art methods. Moreover, GAP-WF can achieve higher accuracy with fewer training samples. These results indicate that GAP-WF can make better use of the fine-grained features in traffic trace for fine-grained website fingerprinting.

ACKNOWLEDGMENT

This work is supported by The National Key Research and Development Program of China(No.2018YFB1800200, No.2020YFB1006100 and No.2020YFE0200500) and Key research and Development Program for Guangdong Province under grant No.2019B010137003 and Defense Industrial Technology Development Program under grant No.JCKY2019211B001.

REFERENCES

[1] A. Freier, P. Karlton, and P. Kocher, "The secure sockets layer (ssl) protocol version 3.0," RFC 6101, Internet Engineering Task Force, August 2011.

[2] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," RFC 5246, Internet Engineering Task Force, August 2008.

[3] W3techs, "Usage statistics of default protocol https for websites." [Online]. https://w3techs.com/technologies/details/ce-httpsdefault, Accessed January 19, 2021.

[4] Z. Zhang, C. Kang, G. Xiong, and Z. Li, "Deep forest with lrrs feature for fine-grained website fingerprinting with encrypted ssl/tls," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 851–860, 2019.

[5] M. Shen, Y. Liu, L. Zhu, X. Du, and J. Hu, "Fine-grained webpage fingerprinting using only packet length information of encrypted traffic," *IEEE Transactions on Information Forensics and Security*, 2020.

[6] T. Wang, "High precision open-world website fingerprinting," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 152–167, IEEE, 2020.

[7] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 1187–1203, 2016.

[8] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *NDSS*, 2016.

[9] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 143–157, 2014.

[10] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 263–274, 2014.

[11] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pp. 201–212, 2013.

[12] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1131–1148, 2019.

[13] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 292–310, 2019.

[14] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1928–1943, 2018.

[15] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," *arXiv preprint arXiv:1708.06376*, 2017.

[16] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Network and Distributed System Security Symposium, NDSS 2020*, Internet Society, 2020.

[17] J. Zheng and D. Li, "Gcn-tc: Combining trace graph with statistical features for network traffic classification," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted http streams," in *International Workshop on Privacy Enhancing Technologies*, pp. 1–11, Springer, 2005.

[20] Z.-H. Zhou and J. Feng, "Deep forest," *arXiv preprint arXiv:1702.08835*, 2017.

[21] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–34, 2012.

[22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.

[23] Contentsquare, "2020 digital experience benchmark." [Online]. https://contentsquare.com/digital-experience-benchmark-hub/.

[24] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[26] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International Conference on Machine Learning*, pp. 3734–3743, PMLR, 2019.

[27] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*, pp. 5453–5462, PMLR, 2018.