IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 11, 2021, accepted May 12, 2021, date of publication June 8, 2021, date of current version June 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3087528

The Challenge of Only One Flow Problem for Traffic Classification in Identity Obfuscation Environments

HONG-YEN CHEN[1] AND TSUNG-NAN LIN[1,2], (Senior Member, IEEE)
[1]Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan

Corresponding author: Tsung-Nan Lin (tsungnan@ntu.edu.tw)

This work was supported in part by the "5G Cyber Security Detection and Protection System Project (1/1)" of the Institute for Information Industry, Taiwan, and in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 110-2634-F-002-037 and Grant 110-2218-E-002-018-MBK.

ABSTRACT As encrypted traffic grows, network flow classification has become a significant issue because of the impossibility to parse the payload in an encrypted packet. A possible packet sniffing location for organizations is an under control gateway between intranet and internet to inspect network traffic. However, when an intranet user uses an identity obfuscation protocol such as VPN or TOR, the packet IP and port would be rewritten to preserve user privacy. The same user's packet sniffed between a user and TOR entry node/VPN proxy always has the same 5-tuples (packets with the same source IP, destination IP, source port, destination port, and IP protocol defined as flow). Thus, we cannot rely on the 5-tuples rule to split traffic into flows. This challenge is called the "only one flow problem" and poses an obstacle for flow classification. A previous solution uses timeout value to determine flow separation points to address this issue. However, the predefined static time threshold cannot fit all user habits, which leads to separation errors. To overcome timeout limitations, we propose a flexible method called AI-FlowDet by leveraging the scene change concept and a CNN model to find behavior change points of traffic based on learning data. AI-FlowDet can apply to the traffic of the identity obfuscation protocols. Next, we propose 294 size-based and direction-based features that can be used with AI-FlowDet to evaluate flow type classification performance. Every experiment leverages different machine learning algorithms. The results show that AI-FlowDet with the proposed features can achieve 98.5% weighted accuracy, which is increased by 12.6% versus the previous timeout method with baseline features. We proved that the proposed splitting methods for the only one flow problem and proposed features for flow type classification are effective based on the good results obtained for both the VPN and TOR datasets.

INDEX TERMS TOR, VPN, flow classification, AI-FlowDet, only one flow problem.

I. INTRODUCTION

With the rise of 5G networks, hacking and information security incidents have also escalated. Security researchers will generally obtain information from endpoint devices (PCs, laptops, mobile phones) or network devices (routers, switches) for inspection. The information obtained from an endpoint device is relatively complete but will involve personal privacy issues, and an agent cannot be installed on all endpoint devices. Therefore, a more feasible method for enterprises is to obtain network traffic from network devices for examination. Current traffic classification studies focus on three domains: traffic application type classification (to determine whether the type of traffic is email, file transfer, or browsing) [1]–[10], website fingerprinting (to distinguish the originated websites or users from specific traffic) [11]–[16], and malicious traffic classification (to detect the traffic generated via hacker/malware and identify its attack type) [17]–[22].

The conventional methods to classify traffic use either a port-based approach [23] or a signature-based approach [24]. In the early stages of the Internet, many applications used a well-known port. However, most recent applications do not use the originally specified port, which causes the port-based method to become unreliable. A signature-based approach, such as deep packet inspection (DPI) [25], relies on signatures

The associate editor coordinating the review of this manuscript and approving it for publication was Junfeng Wang.

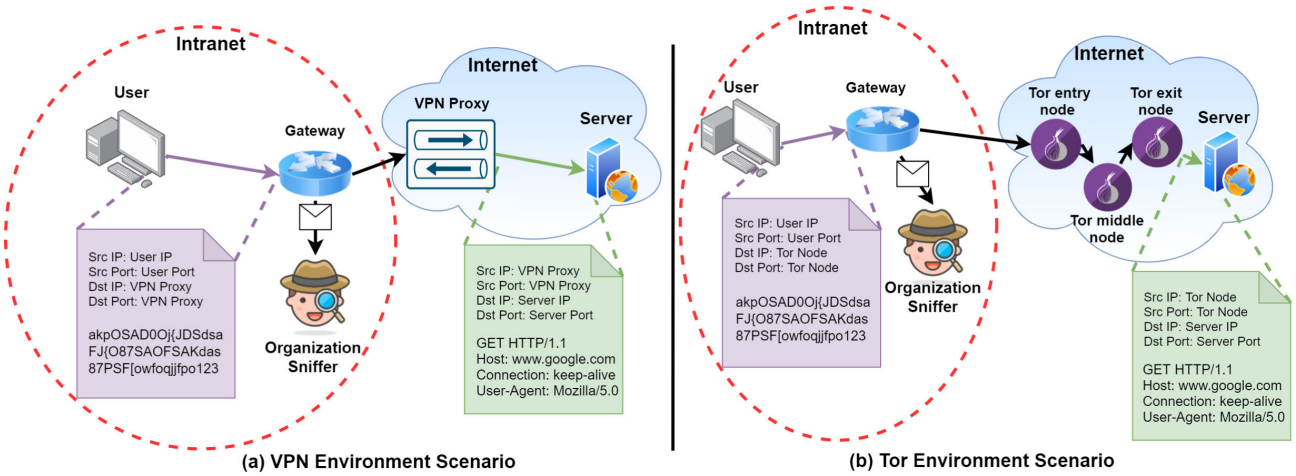


FIGURE 1. The scenarios of VPN (left) and TOR (right) environments. When the intranet user sends an HTTP request packet (without encryption), it will first be encrypted and replace the destination address with a TOR node/VPN Proxy (purple background packet). After the packet is sent to the gateway, the gateway cannot know the server address and only knows the TOR entry node/VPN proxy address. When the packet is sent from the TOR exit node/VPN proxy to the server, the packet is already decrypted, and the source address is the TOR exit node/VPN proxy, not the user address (green background packet). The server would therefore never know the user's identity. When sniffing packets from the gateway, the packets that we obtain all contain the same 5-tuple, which causes the only one flow problem.

of a packet payload (L7). In the case of frequent rule updating, the DPI method can achieve high accuracy and almost no false positives. However, with the rise of network encryption techniques such as TLS and IPSEC, most network traffic has been encrypted, and the payload (L7) is becoming unparseable. Despite using predefined rules, signature-based methods cannot precisely distinguish encrypted traffic. To overcome some of the limitations mentioned above, researchers recently switched to using machine learning-based [1], [2], [5]–[16], [19], [21], [22] or deep learning-based methods [3], [4], [20], achieving excellent performance.

In general, regardless of whether the packet is encrypted or not, the first step is to split traffic into flows by 5-tuples rule (the packets with the same source IP, source port, destination IP, destination port, and IP protocol are defined as a flow). However, some encryption protocols such as Virtual Private Network (VPN, Fig. 1 (a)) or The Onion Router (TOR, Fig. 1 (b)) [26] also offer an effective proxy characteristic, which would replace the source IP and packet port with a proxy address and ensure privacy preservation for the user.

The sniffing packets between the TOR exit node/VPN proxy and the server encounter two obstacles. First, the packet routing path from a TOR exit node/VPN proxy to a server is dynamic, and we cannot sniff 100% of packets from a static router. Second, we cannot precisely know where the TOR node/VPN proxy geographical location is and sniff the packets from the appropriate router. For organizations that want to inspect the network behavior of intranet users, the best passive sniffing packet is a gateway under the organization's control. But when an intranet user leverages identity obfuscation protocols such as VPN or TOR protocols, the traffic becomes inseparable because packets between the same user and a gateway always include the same 5-tuples. We define this challenge as the "only one flow problem", which indicates a flow with infinite length and a connected server to which the packet belongs cannot be distinguished. To solve the problem, previous works [1]–[6] used the timeout value as a terminating flag to split traffic.

However, time intervals between packets are affected by human operation parameters such as mouse click frequency. If the timeout threshold is set as too long, it is easy to misjudge the separation point when traffic is generated by a user with fast application switching and vice versa. In addition, it is impossible to specify a timeout value to satisfy the behavior of all users. To overcome this limitation, we propose a deep learning-based solution named flow detector (AI-FlowDet), which can automatically determine behavior change points of traffic via a learning approach. AI-FlowDet utilizes the concept of image scene change and leverages a 2D convolutional neural network (CNN) to determine the similarity of flow sequences. Unlike the timeout value, which relies on the static threshold, AI-FlowDet depends on knowledge learned from training data to achieve flexibility and independence with respect to the user's habits. We apply the technique to overcome the challenge of the only one flow problem for traffic classification in identity obfuscation environments. Moreover, based on the distinct subflow characteristics of different applications, we propose 294 size-based and direction-based (S&D) features that can be applied to flow type classification.

The objective of this paper is to distinguish the application types of flow in identity obfuscation environments. This includes three steps: (1) applying the proposed deep learning model, named AI-FlowDet, to obtain change points to split traffic into flows, (2) extracting 294 proposed size-based and direction-based features from every flow, and (3) leveraging the machine learning models to classify the application type of every flow. There are two main contributions of this paper:

- We present a method called AI-FlowDet to address the only one flow problem of the identity obfuscation network environment. When leveraging our method in an actual obfuscation environment such as TOR and VPN, compared to the previous timeout value method, AI-FlowDet can increase the accuracy of the flow application type classification by 6.0% under the TOR protocol and by 11.1% under the VPN protocol.
- We propose 294 S&D features that can be extracted from an encrypted flow. When using AI-FlowDet with S&D features for flow application type classification in an identity obfuscation environment, 98.5% accuracy can be achieved with the MLP algorithm. This value is 12.6% higher than that of the previous method using timeout value and baseline features, which can only obtain 85.9% accuracy.

II. BACKGROUND AND RELATED WORKS

In this section, we first describe the characteristics of the identity obfuscation protocol and the problem caused by it. We then illustrate the previous solution called the timeout value to address the only one flow problem. Finally, we introduce the earlier works about flow application type classification in identity obfuscation environments.

A. IDENTITY OBFUSCATION: ENCRYPTION + PROXY

The tunneling protocol is a communications protocol that allows data to be transferred from one network to another. This protocol can be divided into two categories based on the purpose of use, namely, encryption and proxy. GRE and IP in IP are proxy tunneling protocols that encapsulate one IP packet in another IP packet. SSL/TLS belongs to an encrypted tunneling protocol and aims to prevent packet content from being eavesdropped upon. IPSEC is a tunneling protocol with both proxy and encryption features and is often used by VPN. TOR relies on SSL/TLS techniques for encrypting packets and virtual circuit routing for proxy function. Here, we define TOR and VPN as identity obfuscation environments, which simultaneously have encryption and proxy characteristics.

B. THE EFFECT OF IDENTITY OBFUSCATION: THE ONLY ONE FLOW PROBLEM

Under the VPN protocol, every packet from the user will be first sent to the VPN proxy, but not a server. The actual destination IP and port are hidden in the packet by VPN software. After the packet arrives at the VPN proxy, the software will decrypt the packet and obtain the actual destination IP and port. The VPN proxy will then transfer the decrypted packet to the server for the user instead. The full scenario of VPN is illustrated in Fig. 1 (a). From the server's perspective, it will regard the VPN proxy as an actual client and never know the user identity.

TOR is composed of a network based on TOR nodes. Like VPN, TOR software also encrypts the packet and hides the actual destination IP and port. Generally, there will be 3 TOR proxy nodes between the user and the server. The encrypted packet will thus cross 3 proxies and obtain the actual destination address after 3-time decryption. The full scenario of TOR is displayed in Fig. 1 (b).

In summary, when sniffing packets from a gateway in identity obfuscation environments, the destination IP and port of packets are the proxy address rather than the server address. If the proxy is not under our control, we cannot retrieve the actual server address. This poses a challenge (only one flow problem) for classifying the traffic because we lose the basis (5-tuples) for splitting the traffic to flow.

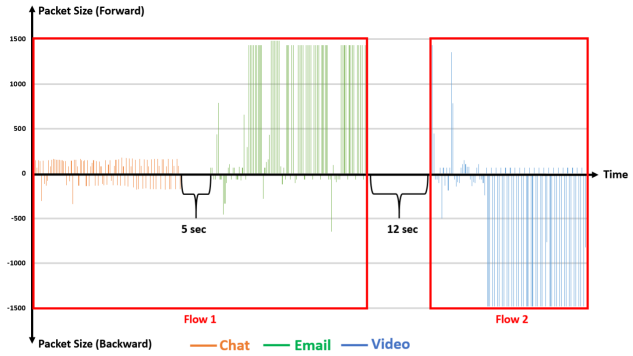


FIGURE 2. The timeout value method for splitting traffic. If we set the timeout as 10 sec, the Chat and Email traffic will be defined as a flow because the time interval between them is not more than 10 seconds. In contrast, the time interval between Email and Video is 12 sec, so the new flow is defined starting from video packets.

C. SPLITTING FLOW METHOD: TIMEOUT VALUE

Previous works [1]–[6] used the timeout value to split the traffic in identity obfuscation environments. The timeout method is based on the inspecting time interval between packets to determine change points. To use this method, we need to set an initial time threshold, such as 10 seconds. If the time interval is longer than the threshold, a junction between packets will be a traffic separation point. Next, the packets that are before the separation point would be considered a flow.

In the real world, every user has different computer use habits. The time interval between packets would be affected by a user's operation. When the timeout threshold is not suitable for the current user, it is easy to incorrectly determine the separation point. As shown in Fig. 2, the timeout threshold is 10 seconds, and the time interval between chat packets and email packets is 5 seconds. The flows of chat and email are thus mistakenly considered a flow. This fault will affect the phase of feature extraction and disrupt the final classification results. In addition, it is difficult to select a static timeout threshold to fit all users. Hence, using the timeout method inevitably causes some misjudgments and harms classification results.

D. FLOW APPLICATION TYPE CLASSIFICATION UNDER AN IDENTITY OBFUSCATION ENVIRONMENT

[6] and [2] focus on the traffic classification in the TOR environment. The earlier work [6] only distinguishes the

traffic into four application types: Web, P2P, FTP, and Others. The authors leveraged the hidden Markov model (HMM) and some base features to perform classification. In their dataset, the probability of a successful guess can reach 92%. The closer work [2] distinguishes the application into eight different types: Browsing, Chat, Video, Audio, Email, VoIP, P2P, or File Transfer. The authors used time-based features (the enhanced version called the CIC flowmeter feature set [27]) to train the machine learning models. When using the random forest algorithm and with the timeout value set to 15 seconds, the best precision is 84.1% and recall is 83.6%.

[1], [3]–[5] focus on traffic classification in the VPN environment. [1] is the first work to distinguish seven different types of VPN traffic: Video, Chat, Audio, Email, VoIP, P2P, and File Transfer. The authors used machine learning with time-based features for flow characterization. The average precision of 84% can be achieved by leveraging decision tree C4.5, and the timeout value is 15 seconds. [3]–[5] used the same dataset provided by [1]. [5] is the closest to [1]. The main differences are that 111 flow-based features are used and that browsing traffic is not considered. In addition, the authors classify the application type and the protocols (Facebook, Skype, and so on). By using the k-nearest neighbors (KNN) machine learning algorithm, 93.84% average accuracy can be achieved. [3] is based on deep learning algorithms. The authors transfer the raw traffic to sequence and try two different models: one-dimensional (1D) convolutional neural network (CNN) and autoencoder. The results show that the CNN is better and can achieve 98% average accuracy. [4] is also based on deep learning, and the authors compared the performance between 1D-CNN and 2D-CNN and found that the accuracy of 1D-CNN is higher than that of 2D-CNN by as much as 2.51%.

[7] focus on the traffic classification of TOR, I2P, and JonDonym via a hierarchical approach with modularity characteristics, training efficiency, distributed deployment, and tunable view of classification outputs. There are three granularities in their structure, namely, the adopted network anonymity (L1), the traffic type tunneled in the network (L2), and the application category in generating such traffic (L3). In applications classification (L3), the proposed method can achieve an F-measure up to 75.56%. [9] is an extension work from [7] and presents a new approach called BDeH, a Big Data-enabled hierarchical framework that capitalizes on two complementary types of parallelism: model and data parallelism. The simulation results show that the proposed method can outperform either pure data or pure model parallelism approaches by reducing training completion time by up to 78% and cloud-deployment costs by up to 31%.

[8] focus on profiling the users and applications on the I2P network. They found that an amount of shared bandwidth would affect profiling the users and the applications. Furthermore, applications that do not use the shared client tunnels increase the possibility of profiting from the flow behavior. When the amount of shared bandwidth is 80%, user profiling can be performed with 81.8% accuracy. [10] proposes a method called packet momentum, which leverages a set of features based on the first 3 packets of a network traffic flow and machine learning to identify multilayer-encryption anonymity networks. Accordingly, 22 application layer protocols in total, such as HTTP, I2P, TOR, and JonDonym, are utilized in experiments. They can achieve 97.92% accuracy.

III. AI-BASED FLOW DETECTOR

This section presents AI-FlowDet, a method to find the application behavior change points in traffic. AI-FlowDet addresses the only one flow problem caused by identity obfuscation environments such as VPN and TOR. Our proposed approach introduces the image scene change detection [28], [29] concept and leverages it on the network traffic. By comparing the context of network packet sequences, AI-FlowDet can accurately calculate the similarity of the packet sequences and determine whether the two sequences are from the same application type or not. If they are from different applications, the junction between two packet sequences would be defined as a behavior change point.

The AI-FlowDet approach relies on the power of data science and is a 2D convolutional neural network (CNN) model [30], [31]. The main point of AI-FlowDet is the training set generation algorithm. To fit the model, we first need to prepare an amount of different network application behavior flows. We then specify three hyper-parameters called contiguous packet threshold (CPT), confidence region (CR), and kernel region (KR) for training. The experimental results of three hyper-parameters are in Section V-A. Finally, we obtain the change points from model prediction and leverage them to split traffic into flows. In detail, the construction of AI-FlowDet can be summarized in four main steps, which are as follows.

(A) **Collecting different network application behavior flows.** In this step, we need to collect specific amounts of regular traffic (not in TOR or VPN) with different application behaviors and only one behavior simultaneously. We can then use the 5-tuples rule to split the traffic into flows. Please refer to additional details in Section III-A.

(B) **Building a packet sequences pair dataset for training.** In this step, we first need to determine a hyper-parameter called the contiguous packet threshold (CPT), which is the packet sequence size extracted from network flows. Based on the CPT, every flow would generate many packet sequence pairs. We then need to specify another hyper-parameter called the confidence region (CR) to determine whether the packet sequence pair is a positive sample (not similar) or a negative sample (similar). See more details in Section III-B.

(C) **Training the 2D CNN model to determine change points.** In this step, we use the packet sequence pairs generated from the previous step to train a CNN model. The CNN model input is a matrix that is composed of

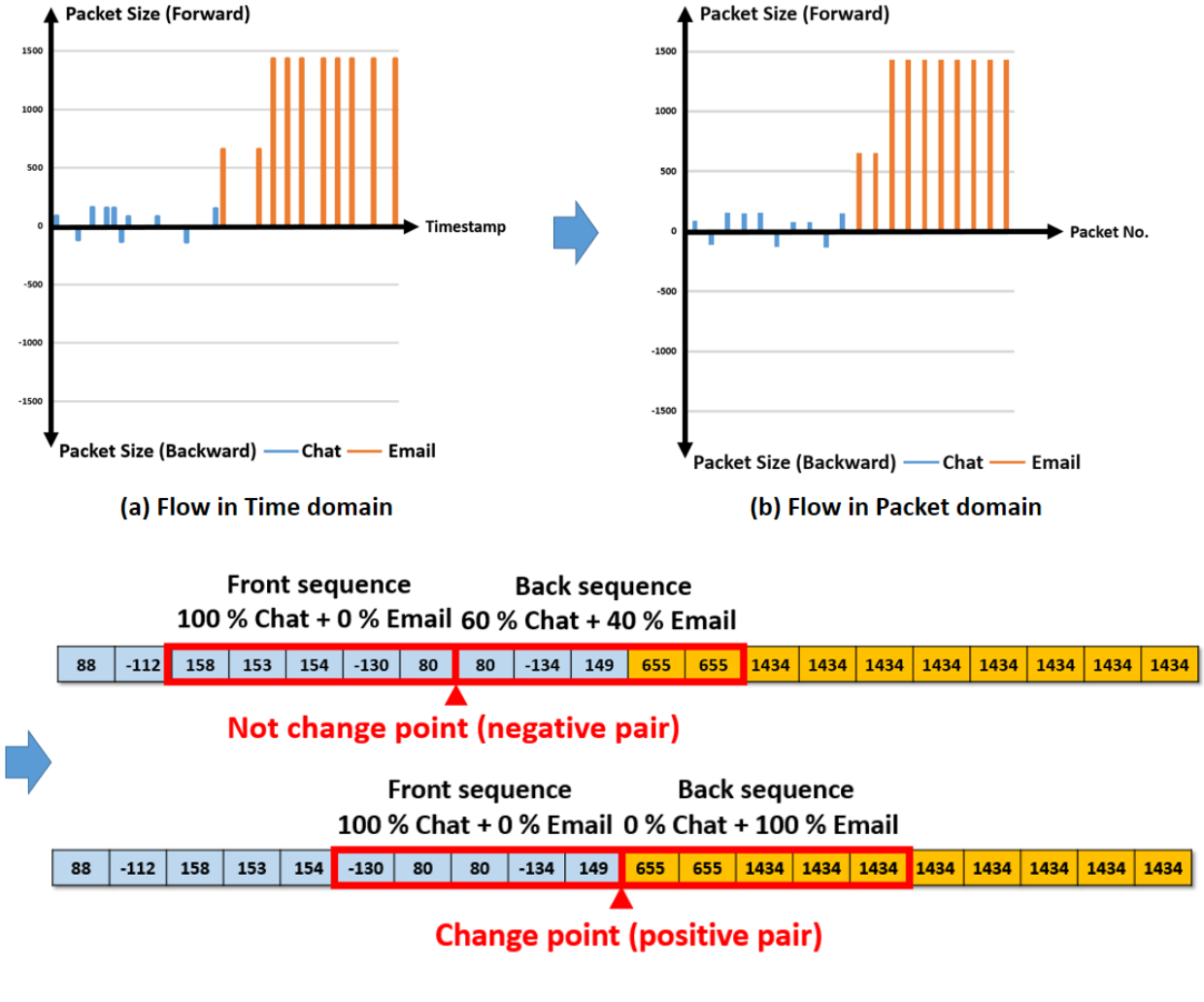


FIGURE 3. The procedure of training dataset generation. (a) shows the flow in the time domain. The blue and green indicate chat and email flow. The forward packet is a positive value, and the backward packet is a negative value. (b) shows the flow projecting from a time domain to a packet domain and becoming a packet sequence. The intervals between packets are not based on time. (c) shows the definition of sample pairs. The CPT value here is 5, and CR is 50%. The top sequence pair is a negative sample pair and the bottom sequence pair is a positive sample pair.

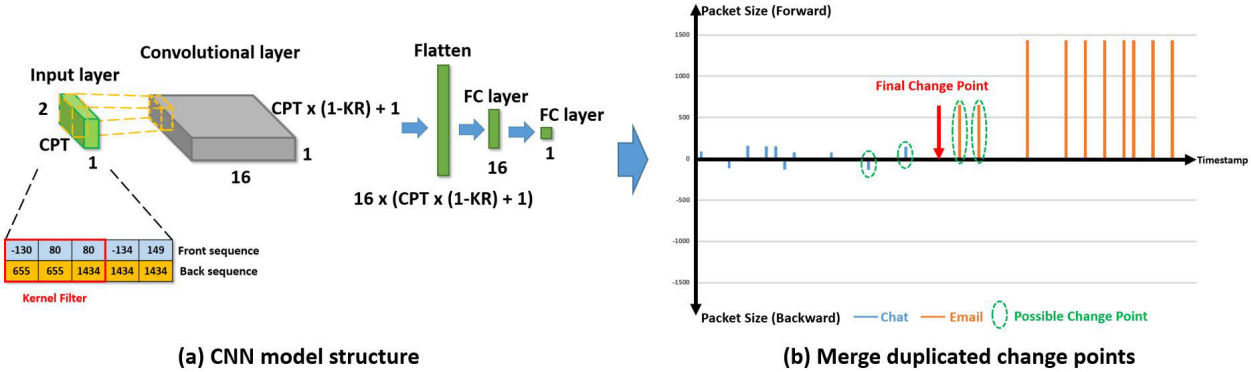


FIGURE 4. The procedure for finding splitting points. (a) presents a CNN model structure. The input is a matrix composed of 2 sequences, and the output is possibility of change points. (b) shows the merging procedure. We first cluster the close change points into a group and average their timestamps to generate final change points.

two packet sequences. A CNN model's output is a binary value, for which 0 means that the two packet sequences are from the same flow and 1 means that they are from different flows. Here, we would investigate the impact of using different kernel sizes (KR) in the convolutional layer. Please refer to additional details in Section III-C.

TABLE 1. The packet statistics of the dataset (the results are the average of 20 flows).

	Packet Number	Byte Number	Byte/Packet
Audio	2921	2.3 MB	787
Browse	5566	4.3 MB	773
Chat	788	0.3 MB	505
Email	1402	1.6 MB	1193
FT	51743	60.3 MB	1174
P2P	14377	9.5 MB	700
Video	48679	45.7 MB	925
VoIP	63670	10 MB	216

(D) **Merging the duplicated change point and splitting the traffic into flows.** In this step, we leverage the model on the traffic with only one flow problem. The classification results show the locations of change points. Before splitting traffic, we need to merge the change points according to the confidence region (CR) to eliminate the duplicated points. After merging, traffic can be split into flows by new change points. See more details in Section III-D.

A. COLLECTING DIFFERENT NETWORK APPLICATION BEHAVIOR FLOWS

The UNB dataset [1], [2] is composed of 2 parts: one is regular traffic (not in VPN/TOR), and the other is VPN/TOR traffic. It is available for academic usage. We leverage regular traffic in this section, from which it is easy for us to obtain ground truth to prepare training data. There are eight different application behaviors: Browsing, Chat, Video, Audio, Email, VoIP, P2P, and File Transfer. Finally, we choose 20 flows from each application type and a total of 160 flows for the next step. The statistical details of the dataset are shown in Table 1.

B. BUILDING A PACKET SEQUENCES PAIR DATASET FOR TRAINING (TIME DOMAIN TO PACKET DOMAIN)

This step's core is using the flows from Section III-A to generate the packet sequence pairs and ground truth labels. The packets in the time domain are as shown in Fig. 3 (a), where the x-axis indicates a timestamp, and the y-axis displays packet length. In Fig. 3 (a), the packet's coordinates and interval distance are based on the influence of time. Here, however, we do not need to leverage the time series concept. We decouple it and only retain the order concept of packets. The decoupled result is shown in Fig. 3 (b), where the x-axis only displays packet order but not timestamp, and the y-axis still displays packet length. The packet sequence is a part of a flow for which length is determined by the hyper-parameters, which is called the contiguous packet threshold (CPT, sliding window size). The sequence's value is packet size, for which positive indicates the forward direction and negative indicates the backward direction. Here, different CPT assume the shortest length of each flow that we will split from traffic. If the CPT is set as too long, a packet sequence would have a higher possibility with two more application types. This situation would confuse the CNN model in the training stage. To avoid more than two flows in a sliding window, we determine the CPT based on the average packet number of flows counted in Table 1. The result shows that most flow has more than 780 packets. So we select a number less than 780, such as 100, 200 for experiments. But in the real world, it still has short flows in traffic, so we discuss the issue in Section V-G. After specifying the CPT, we select two different application type flows and concatenate them together. The packet sequence window, which is the CPT size, would then slide from the beginning to the end of the merged flow. Next, we need to specify a hyper-parameter called the confidence region (CR) to determine whether the front and back packet sequences represent a positive and negative sample pair. The CR is a percentage threshold value; it represents when the proportion of two sequences is composed of different flows. The definition of a sequence pair example is shown in Fig. 3 (c). The labeling details are found in Algorithm 1.

TABLE 2. The 2D CNN Structure of Flow Scene Change Detection.

Type / Stride	Filter Shape	Input Size
Conv / s1	2 x 100 x 16	2 x 200 x 1
ReLU	-	1 x 101 x 16
Flatten	-	1 x 101 x 16
FC / s1	1616 x 16	1616
ReLU	-	16
FC / s1	16 x 1	16
Sigmoid	-	1

*CPT = 200, KR = 0.5
*Filter size = CPT x KR = 100

C. TRAINING THE 2D CNN MODEL TO DETERMINE CHANGE POINTS

This step would use the sequence pairs and labels from Section III-B to train a 2D CNN model. As shown in Fig. 4 (a), the input of the 2D CNN is a matrix, and the size is 2 * CPT. The first row of the matrix is the front packet sequence, and the second row is the back packet sequence. Every value in the matrix indicates every packet length, where a positive value indicates an outbound (sent) packet, and a negative value denotes an inbound (received) packet. The proposed CNN model can thus be based on packet length and direction to determine the splitting point. The first layer is a convolutional layer with a ReLU activation function, and the size of the filter is 2 * N, where N equals KR * CPT. The KR is a hyper-parameter, and its range is from 0-1. It impacts the model of how to compare the packet sequences. If the KR is longer, the model would compare more packets simultaneously, and vice versa. The second and last layers are a fully connected layer and then using a sigmoid activation to normalize the output value to the range of 0-1, where 1 represents the change point and 0 represents the non-change point. The structure details are shown in Table 2.

D. MERGING THE DUPLICATED CHANGE POINT AND SPLITTING THE TRAFFIC INTO FLOWS (PACKET DOMAIN TO TIME DOMAIN)

After training a CNN model, we leverage it on traffic, which has only one flow problem. We then obtain all change points of the traffic. Here, however, we will not directly use change

Algorithm 1 Training Set Label

INPUT:
CPT: Contiguous Packet Threshold (Integer)
CR: Confidence Region (%)
M: Length of flow 1 (Integer)
N: Length of flow 2 (Integer)
OUTPUT:
L: List of labels. (Bool, True is change point)
Initialization:
$L \leftarrow []$
for each $i \in [CPT, M + N - 2 * CPT - 1]$ **do**
  if $i < M - CPT * (2 - CR)$ **then**
    $L.append(False)$
  else if $i > M - CPT * CR$ **then**
    $L.append(False)$
  else
    $L.append(True)$
  end if
end for

Algorithm 2 Merge Duplicated Points

INPUT:
PTS: List of Packet TimeStamps (Float)
CP: List of Change Points (Integer)
CPT: Contiguous Packet Threshold (Integer)
OUTPUT:
$NCPTS$: List of New Change Point TimeStamps (Float)
Initialization:
SG: Packet is now in the Same Group or not (Bool)
GM: List of Member in Group (Integer)
$GMTS$: List of TimeStamps of Group Member (Float)
$SG \leftarrow False$
$GM \leftarrow []$
$NCPTS \leftarrow []$
for each $i \in [0, len(CP)]$ **do**
  if $SG == False$ **then**
    $GM.append(CP[i])$
    $SG \leftarrow True$
  else
    if $CP[i+1]$ - $CP[i] < CPT$ **then**
      $GM.append(CP[i])$
    else
      $GMTS \leftarrow []$
      for each $j \in [0, len(GM)]$ **do**
        $GMTS.append(TS[j])$
      end for
      $AveTimeStamps \leftarrow mean(GMTS)$
      $NCPTS.append(AveTimeStamps)$
      $SG \leftarrow False$
      $GM \leftarrow []$
    end if
  end if
end for

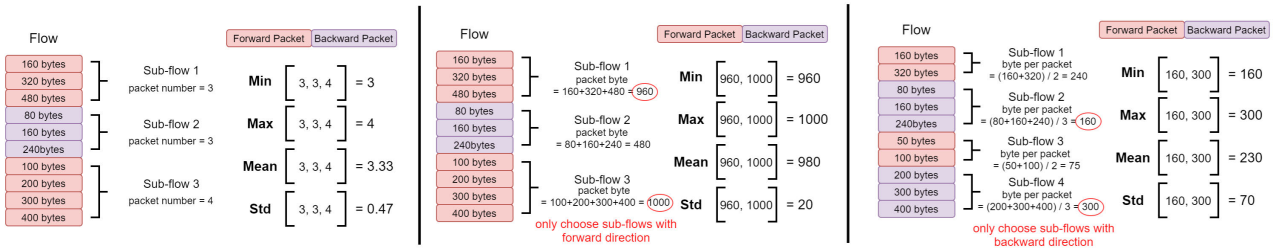
points because the hyper-parameter CR used in the training phase causes duplicated change points in the testing phase. In detail, the definition of a positive sample is affected by the CR. If the CR value is lower, the change point can be next to the junction instead of just at the intersection. Thus, we need to merge the duplicated points in the testing phase. Points not exceeding the length of the CPT will be clustered into a group. We then reproject the packets from the packet domain to the time domain. As shown in Fig. 4 (b), we average each point's timestamps in the same group and leave only the closest average timestamp point. Finally, every group would generate a new change point. The traffic would then be based on new change points to split into flows. The packets in a flow exhibit the same application behavior. The details for merging change points is found in Algorithm 2.

This section illustrates how to obtain the change points of traffic by the proposed method and how to use them to determine flows. This method is divided into two steps in actual use. The first step is to find splitting points. If the number of packets is N, then O(N-CPT+1) CNN model operations are required. The second step is to merge the splitting points. The computational resources required in this step will be determined according to the number of splitting points (N) found in the previous step, and the average computational complexity is O(2N). Next, to classify each flow's application type, we propose 294 new features which can be extracted from flow whether encrypted or not.

IV. S&D: SIZE-BASED AND DIRECTION-BASED FEATURES

In this section, we present the size-based and direction-based (S&D) features for flow application type classification. The core of S&D is to measure the packet size in a subflow, which is composed of consecutive packets in the same direction. The size and direction of a packet generated by different network applications will be distinct, e.g., in the file transfer traffic, most packets have the same direction, and the packet size is almost the maximum transmission unit (MTU). We can thus observe that the file transfer traffic contains few subflows and the packet size in subflow exhibits a high average value and low standard deviation. On the other hand, in chat traffic, packets' directions are uniformly distributed, and the direction changes frequently. The subflow number of chat traffic would exceed that of file transfer, and the packet size in subflow presents a low average. Hence, the characteristics of subflow can enable machine learning models to more accurately distinguish different types of flow. The following list describes a total of 294 features we proposed.

- Fwd2Bwd: The time of packet direction change from forward to backward (one feature).
- Bwd2Fwd: The time of packet direction change from backward to forward (one feature).
- Statistical values of the subflow packet number: The minimum, maximum, mean and standard deviation values of the total packet number among all subflows with



(a) Statistical values of sub-flow packet number (b) Statistical values of forward sub-flow packet bytes (c) Statistical values of backward sub-flow bytes per packet

FIGURE 5. Illustration of proposed features.

both directions. An example is shown in Fig. 5 (a) (four features).

- Statistical values of the forward subflow packet number: The minimum, maximum, mean and standard deviation values of total packet number among all subflows with a forward direction (four features).
- Statistical values of the backward subflow packet number: The minimum, maximum, mean and standard deviation values of total packet number among all subflows with a backward direction (four features).
- Statistical values of subflow packet bytes: The minimum, maximum, mean and standard deviation values of the total packet size among all subflows with both directions (four features).
- Statistical values of forward subflow packet bytes: The minimum, maximum, mean and standard deviation values of the total packet size among all subflows with a forward direction. An example is shown in Fig. 5 (b) (four features).
- Statistical values of backward subflow packet bytes: The minimum, maximum, mean and standard deviation values of the total packet size among all subflows for the backward direction (four features).
- Statistical values of subflow bytes per packet: The minimum, maximum, mean and standard deviation values of the total bytes per packet values among all subflows averaged for both directions (four features).
- Statistical values of forward subflow bytes per packet: The minimum, maximum, mean and standard deviation values of the total bytes per packet values among all subflows averaged for the forward direction (four features).
- Statistical values of backward subflow bytes per packet: The minimum, maximum, mean and standard deviation values of the total bytes per packet values among all subflows averaged for the backward direction. An example is shown in Fig. 5 (c) (four features).
- Byte distribution per packet: Every byte distribution per packet in one flow (from bytes 0 to 255 and total 256 features).

V. EXPERIMENTS ON FINDING FLOW SPLITTING POINTS AND FLOW APPLICATION TYPE CLASSIFICATION

This section first shows the results of finding flow splitting points based on the proposed AI-FlowDet. Next, we introduce the traffic data of the TOR and VPN environments. Finally, we show the flow type classification results on different feature sets with two flow splitting methods.

TABLE 3. F1-score of different CPT, CR and KR.

			Confidence Region (CR)		
			0.25	0.50	0.75
100 CPT	Kernel Region (KR)	0.25	97.2 ± 1.0	97.7 ± 0.5	97.1 ± 1.2
		0.50	97.6 ± 0.6	97.2 ± 0.6	96.7 ± 1.1
		0.75	97.0 ± 0.8	96.4 ± 1.0	96.7 ± 0.9
200 CPT	Kernel Region (KR)	0.25	98.2 ± 0.5	98.0 ± 0.9	**98.4 ± 0.9**
		0.50	98.2 ± 0.7	98.2 ± 0.7	98.1 ± 0.6
		0.75	98.1 ± 0.5	98.1 ± 0.7	98.2 ± 0.9

A. RESULTS OF AI-FlowDet ON FINDING FLOW SPLITTING POINTS

In this experiment, we leverage the regular traffic (not in TOR/VPN) with different application behaviors for training AI-FlowDet and evaluate the performance of finding flow splitting points. As mentioned in Section III, there are three hyper-parameters called contiguous packet threshold (CPT), confidence region (CR), and kernel region (KR) that would affect performance. Here, we show the results on Table 3 which has 18 different hyper-parameter combinations (CPT are 100 and 200. CR are 0.25, 0.5, and 0.75. KR are 0.25, 0.5, and 0.75.). The first three rows are the F1-score results of 100 CPT on different CR and KR, and the rest of the rows are F1-score results of 200 CPT. As one can see, the average F1-score of CPT 100 is 97.1%, and the standard deviation is 0.9%. In CPT 200 results, the average F1-score is 98.1% which is higher than CPT 100, about 1%, and the standard deviation is 0.7%. Further, the best performance of CPT 200 is 98.4% F1-score when CR equals 0.75 and KR equals 0.25. The result shows that AI-FlowDet can find the appropriate flow segmentation point and is robust because of the low standard deviation it obtains. In the next stage of classifying traffic types, we will use the best hyper-parameter combination (CPT is 200, CR is 0.75, and KR is 0.25).

B. DESCRIPTION OF FLOW APPLICATION TYPE DATASETS: TOR AND VPN

Since AI-FlowDet is for identity obfuscation but is trained on regular traffic, it is important to confirm that the packet characteristics of the identity obfuscation environment are not

TABLE 4. Application types included in each category.

Category	Applications
Browse	HTTP or HTTPS traffic from Firefox and Chrome Browser.
Email	Thunderbird and Gmail by using SMTP/S and POP3/SSL.
Chat	Facebook and Hangouts via web browser. Skype, IAM ICQ via an application called pidgin.
Audio	Music streaming from Spotify.
Video	Video streaming from Youtube and Vimeo.
FT	Skype file transfers, FTP over SSH (SFTP) and FTP over SSL (FTPS).
VoIP	Voice-calls from Facebook, Hangouts and Skype.
P2P	File-sharing protocols from BitTORrent.

significantly different from those of the regular environment. Here, we recorded the traffic in the TOR and non-TOR environments at the same time and conducted the same for the VPN and non-VPN environments. The ratios of packet number per sec, packet bytes per sec, and bytes per packet between TOR and non-TOR are 7.5%, 7.3%, and 15.7%, respectively. The ratios of packet number per sec, packet bytes per sec, and bytes per packet between VPN and non-VPN are 0.3%, 11.2%, and 10.3%, respectively. The results show that the packet characteristics between regular and TOR/VPN environments are similar.

As mentioned in Section III-A, the UNB dataset [1], [2] includes regular traffic and VPN/TOR traffic. In this section, we leverage the VPN/TOR traffic for the flow type classification problem. The collected traffic between users and gateways is utilized for different types of applications with the assumption that a user would not use more than one application simultaneously. Thus, during traffic recording, one application is executed at a time. All of the traffic was encrypted by the TOR environment and with the same 5-tuples. After collection, all applications were divided into the eight categories mentioned in Section II-D. The application behavior details about every category are shown in Table 4. The VPN traffic capture procedure is the same as TOR but has only the seven application types mentioned in Section II-D.

C. RESULTS OF FLOW APPLICATION TYPE CLASSIFICATION IN THE TOR ENVIRONMENT

In this experiment, we first compare the two packet splitting methods (Timeout vs. AI-FlowDet) on the TOR traffic dataset mentioned in Section V-B. Based on previous works, the best timeout threshold is 15 seconds, and we also select the same value for experiments. In the AI-FlowDet, we choose the following hyper-parameters obtained from Sec 3-A: CPT of 200, CR of 0.75, and KR of 0.25. Next, we compare the 294 proposed features with the baseline features called CIC flowmeter [27]. CIC flowmeter includes 80 features, including timestamp, IP, port, time-based features, packet-based features, TCP flag, and so on. To avoid overfitting, we removed the timestamp and 5-tuple from CIC flowmeter and ultimately used 74 features for experiments. In each experiment, we leveraged five different machine learning algorithms: k-nearest neighbors (KNN), decision tree (DT), random forest (RF), LightGBM (LG), and multilayer perceptron (MLP).

TABLE 5. Weighted accuracy in TOR dataset.

	Timeout CICflowmeter	AI-FlowDet CICflowmeter	AI-FlowDet Proposed	AI-FlowDet Mixed
LG	87.0 ± 4.1	95.8 ± 0.7	98.7 ± 0.2	98.9 ± 0.3
DT	78.1 ± 3.5	92.3 ± 0.5	94.7 ± 0.4	94.9 ± 0.5
RF	82.9 ± 3.3	94.8 ± 0.6	97.5 ± 0.4	97.7 ± 0.5
KNN	72.3 ± 2.7	87.9 ± 0.6	93.1 ± 0.5	92.0 ± 0.6
MLP	88.9 ± 4.2	94.9 ± 0.6	**99.7 ± 0.1**	99.0 ± 0.1

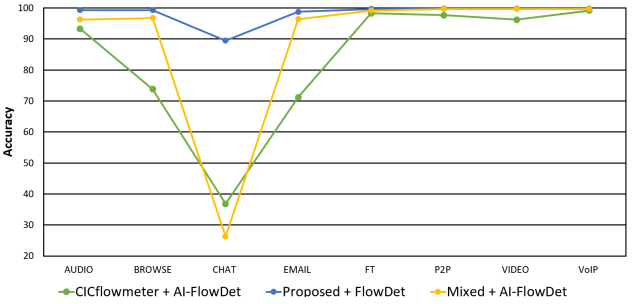


FIGURE 6. Accuracy of every class in the TOR environment. "Mixed" here indicates the set of features, including the CIC flowmeter and proposed features.

TABLE 6. Weighted accuracy in the VPN dataset.

	Timeout CICflowmeter	AI-FlowDet CICflowmeter	AI-FlowDet Proposed	AI-FlowDet Mixed
LG	87.3 ± 2.2	98.4 ± 0.5	97.9 ± 0.6	**98.8 ± 0.5**
DT	81.0 ± 2.1	94.6 ± 0.7	94.2 ± 1.2	95.5 ± 0.7
RF	86.1 ± 2.3	97.3 ± 0.4	95.8 ± 0.2	97.7 ± 0.4
KNN	74.5 ± 2.3	91.8 ± 0.7	90.3 ± 0.9	93.1 ± 0.5
MLP	82.9 ± 2.5	96.8 ± 0.5	97.3 ± 0.3	97.8 ± 0.7

We use weighted accuracy as an evaluation metric and in five-fold cross-validation, and experimental results are shown in Table 5. The first and second columns compare the AI-FlowDet and Timeout flow splitting methods for the same feature set. As one can see, our approach shown in the second column achieves better performance for every algorithm. The best algorithm for the timeout method is MLP, which can achieve 88.9% weighted accuracy. In comparison, AI-FlowDet can attain 94.9% weighted accuracy, which is 6% better than that of the CIC flowmeter.

The results of the second, third, and fourth columns are for evaluating different feature sets with the same splitting flow method. As the second and third columns show, our features offer better performance than those of the CIC flowmeter for each algorithm. Moreover, we try combining two feature sets to improve performance. However, the results of the third and fourth columns are similar, which implies that our features include the characteristics of the CIC flowmeter features in the TOR environment.

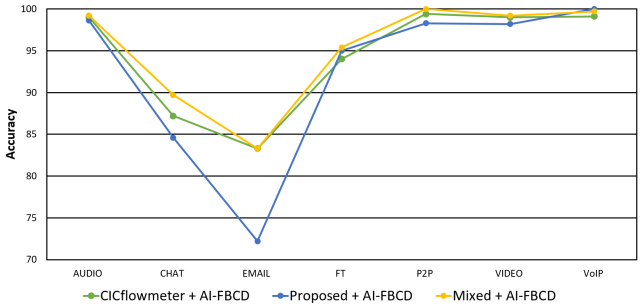


FIGURE 7. Accuracy of every class in the VPN environment. "Mixed" here indicates the set of features, including the CIC flowmeter and proposed features.

D. RESULTS OF FLOW APPLICATION TYPE CLASSIFICATION IN THE VPN ENVIRONMENT

In this experiment, we focus on the VPN traffic dataset mentioned in Section V-B, and the settings and algorithms are the same as in Section V-C. Based on Table 6, we can see that the outcome is similar to the results in the TOR environment. Based on columns one and two, our AI-FlowDet method outperforms the timeout method with the same features by 11.1% under the LightGBM algorithm. However, in the VPN environment, the difference in performance between the two feature sets is not significant. When we merge two feature sets, we can achieve the best performance of 98.8% weighted accuracy with LightGBM.

E. FURTHER INVESTIGATION OF FEATURE SETS AND COMPREHENSIVE COMPARISON

To deeply analyze the effects of features, we show the performance of every class in TOR (in Fig. 6) and VPN (in Fig. 7) environments. In TOR, we can see that AI-FlowDet with the proposed features always offers better accuracy in every class, especially in chat, where it increases accuracy by 50%. Nevertheless, after merging the CIC-flowmeter feature set, the accuracy of some classes slightly declines, while that of chat decreases dramatically. In VPN, the CIC flowmeter has better performance than proposed features in most classes. Unlike TOR results, when we combine two feature sets, the accuracy is better than when using only one of the feature sets.

Next, we present a comprehensive comparison of two features in the identity obfuscation environment's average results and all ML algorithms. The accuracy of the CIC flowmeter is 94.4%, including values of column 2 of Table 5 and Table 6. On the other hand, the accuracy of proposed features is 95.9%, including values of column 3 of Table 5 and Table 6. The 1.5% accuracy improvement confirms that our features are more robust than the CIC flowmeter. Moreover, this result implies that when it is impossible to determine whether the identity obfuscation environment is VPN or TOR, leveraging our features has a high probability of achieving better performance.

TABLE 7. Feature importance rank. (The bold means proposed feature).

Rank	VPN Dataset	TOR Dataset
1	**byte_78_per_packet**	std_pkt_len
2	mean_fwd_pkt_len	var_pkt_len
3	max_pkt_len	**mean_sflow_byte_num**
4	**byte_0_per_packet**	**byte_3_per_packet**
5	**byte_224_per_packet**	**std_sflow_byte_per_pkt**
6	**byte_63_per_packet**	**mean_sflow_pkt_num**
7	init_bwd_win_bytev	**byte_197_per_packet**
8	ack_flag_cnt	max_biat
9	**byte_233_per_packet**	**mean_fwd_sflow_byte_per_pkt**
10	fwd_seg_size_ave	max_flowiat
11	**std_sflow_pkt_num**	**byte_25_per_packet**
12	**byte_255_per_packet**	**byte_33_per_packet**
13	max_active	**mean_sflow_byte_per_pkt**
14	**byte_16_per_packet**	std_flowiat
15	**byte_111_per_packet**	**byte_201_per_packet**
16	**max_sflow_byte_per_pkt**	**std_fwd_sflow_byte_per_pkt**
17	max_bwd_pkt_len	**byte_29_per_packet**
18	init_fwd_win_byte	max_fiat
19	duration	**mean_bwd_sflow_byte_per_pkt**
20	**byte_31_per_packet**	**byte_170_per_packet**
21	fb_psec	**byte_182_per_packet**
22	min_flowiat	std_bwd_pkt_len
23	**byte_144_per_packet**	**byte_200_per_packet**
24	bwd_PSH_flags_number	**byte_74_per_packet**
25	mean_flowiat	**byte_87_per_packet**

F. FEATURE IMPORTANCE AND RANKING

Due to the high number of features (294+74) used for ML algorithms, in this part, we obtain the importance value of every feature as based on the LightGBM algorithm and show which features are truly informative. LightGBM is a gradient boosting framework that uses tree-based learning algorithms; it finds a feature with the highest information gain to be a tree splitting point in every iteration. The more information gain a feature has, the more important is that feature. After training the LightGBM model, we can obtain every feature's information gain. The top 25 important features in TOR and VPN are shown in Table 7. It can be found that 18 of our proposed features are shown in the TOR dataset, and 12 of our proposed features are shown in the VPN dataset. In addition, we show the results under the different feature numbers in Fig. 8. When the number of features reaches 40, the accuracy is saturated and increases only slowly. The above results prove that proposed features can bring about significant effects for ML algorithms and are informative.

G. NOISE IMPACT FOR AI-FlowDet

As mentioned in Section III-B, we assume the CPT is the shortest length of each flow splitting from traffic. But there may be a short flow mixed with traffic. It will cause more than two flows in a sliding window and impact the AI-FlowDet splitting results. Here, we simulate the situation of one short flow (200 packets) mixing between two long flows (more than 500 packets) and leverage trained AI-FlowDet to find splitting points. The experiments show that when the CPT of AI-FlowDet is 200, AI-FlowDet will consider the short flow part as many splitting points (As Fig. 9 (a)). After leveraging the proposed merge algorithm, we finally obtain a

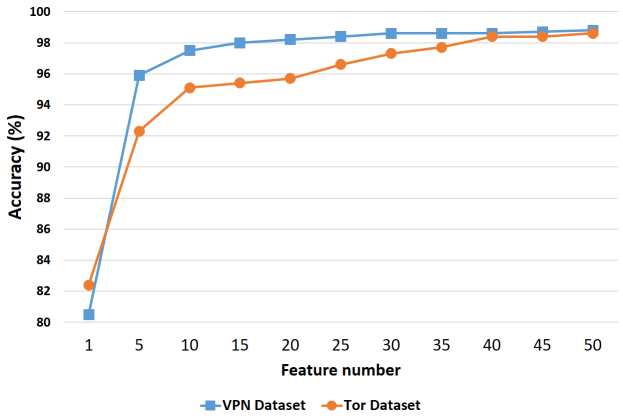


FIGURE 8. The results for different feature numbers. We gradually increase the number of features according to the importance of features and test each feature set. The blue line denotes the results for the VPN dataset, and the orange line denotes the results for the TOR dataset. As shown in the figure, when the number of features reaches 40, the accuracy is saturated and increases slowly.

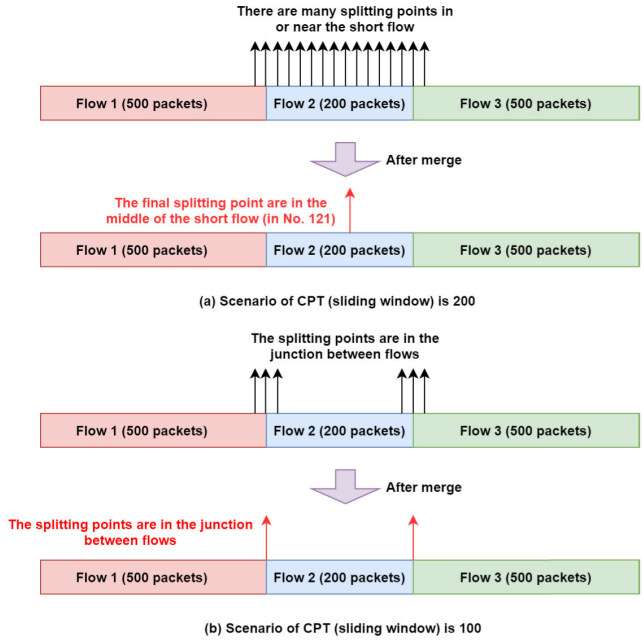


FIGURE 9. The result of AI-FlowDet finding splitting points under the short flow condition. (a) shows that when CPT is not smaller than the length of short flow, AI-FlowDet will consider the short flow part as many splitting points. (b) shows that when CPT is smaller than the length of short flow, AI-FlowDet will correctly find the spitting points.

splitting point in the middle of the short flow part (the average position is on the No. 121 packet of short flow). On the other hand, when the CPT of AI-FlowDet is 100 (half of the short flow), AI-FlowDet will correctly find the two splitting points: the head and the end of the short flow. (As Fig. 9 (b)). In summary, the AI-FlowDet ignores the short traffic, which length is longer than the CPT, and split it into two parts for front-flow and back-flow.

Since AI-FlowDet ignores short flow and integrates it as noise packets into other flows, we design an experiment

TABLE 8. The classification results between with noise packet or not under different classidiers.

	VPN Dataset		TOR Dataset	
	original accuracy	adding noise accuracy	original accuracy	adding noise accuracy
LGBM	98.8 ± 0.5	94.2 ± 0.6	98.9 ± 0.3	98.5 ± 0.1
DT	95.5 ± 0.7	88.6 ± 0.9	94.9 ± 0.5	88.5 ± 6.4
RF	97.7 ± 0.4	94.4 ± 0.7	97.7 ± 0.5	97.5 ± 0.4
KNN	93.1 ± 0.5	92.6 ± 0.1	92.0 ± 0.6	91.7 ± 0.2
DNN	97.8 ± 0.7	97.3 ± 0.1	99.0 ± 0.1	98.3 ± 0.4

to investigate the impact of flow type classifier's when the noise packet is concatenated after the normal flow. The noise packet number we use is 100 (in CPT 200 condition, a flow will be concatenated 100 noise packets on average), and the comparison results are shown in Table 8. Columns one and two are the comparison result with noise or not under the VPN dataset. As you can see, there is no significant difference (the average gap is 3.2%) between noise with or not. The classifier with the most critical gap is the decision tree (about 6.9%), and MLP has the smallest gap (about 0.5%). Columns three and four are under TOR, and the results are similar to VPN(the average gap is 1.6%). The classifier with the most critical gap is the decision tree (about 6.4%), and RF has the smallest gap (about 0.2%). When we leverage our proposed method in real traffic, the performance would be based on the short flow appearing ratio. In summary, the above results show that our method is robust and has little impact on adding noise packets.

VI. CONCLUSION

With the growing awareness of information security, the issue of traffic classification has become more important day by day. Because of endpoint device limitations and privacy issues, network traffic can usually only be obtained from network devices. When network traffic is generated from an identity obfuscation environment, it causes the only one flow problem, and we cannot leverage the 5-tuple to split traffic into flows. To address this issue, we leverage the scene change detection concept to propose a deep learning-based flow splitting method called AI-FlowDet. Under the flow application type classification, compared to previous work using the timeout value to split traffic, our method can increase the accuracy by 6.0% in the TOR dataset and by 11.1% in the VPN dataset. Moreover, we design 294 size-based and direction-based features based on the characteristics of application type. The flow type classification experiments show that the best performance in TOR is achieved when combining AI-FlowDet with proposed features, which can achieve 99.7% weighted accuracy in the MLP algorithm. On the other hand, in VPN, we need to merge the previous feature set, called CIC flowmeter, into AI-FlowDet and the proposed features, through which a weighted accuracy of 98.8% can be obtained in the LightGBM algorithm. According to the great performance obtained in both identity obfuscation environments, we proved that AI-FlowDet and the proposed

features are superior to the previous work with respect to flow classification.

VII. LIMITATIONS AND FUTURE WORK

In the training dataset, only one application type was recorded at the same time. In the actual field, however, various application types will be mixed. Our method can thus only estimate the most significant application behavior of each segment flow. How to accurately split each flow in TOR remains a problem (the VPN environment would face the same problem). In future work, we will more deeply investigate the identity obfuscation protocol environment and find a robust rule to split flow. We will then combine this with the features we proposed to enable this algorithm to be implemented in noisy traffic environments.

REFERENCES

[1] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.

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

[3] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, Feb. 2020.

[4] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 43–48.

[5] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2017, pp. 843–848.

[6] G. He, M. Yang, J. Luo, and X. Gu, "Inferring application type information from Tor encrypted traffic," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, Nov. 2014, pp. 220–227.

[7] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescape, "A dive into the dark Web: Hierarchical traffic classification of anonymity tools," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1043–1054, Jul. 2020.

[8] K. Shahbar and A. N. Zincir-Heywood, "Effects of shared bandwidth on anonymity of the I2P network users," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2017, pp. 235–240.

[9] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A big data-enabled hierarchical framework for traffic classification," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2608–2619, Oct. 2020.

[10] K. Shahbar and A. Zincir-Heywood, "Packet momentum for identification of anonymity networks," *J. Cyber Secur. Mobility*, vol. 6, no. 1, pp. 27–56, 2017.

[11] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at Internet scale," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Feb. 2016, pp. 1–15.

[12] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in *Proc. 12th ACM Workshop Workshop Privacy Electron. Soc.*, Nov. 2013, pp. 201–212.

[13] T. Wang and I. Goldberg, "On realistically attacking Tor with website fingerprinting," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 4, pp. 21–36, Oct. 2016.

[14] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc. (WPES)*, 2011, pp. 103–114.

[15] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable Website fingerprinting technique," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 1187–1203.

[16] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1830–1843, Aug. 2017.

[17] J. Wang, L. Yang, J. Wu, and J. H. Abawajy, "Clustering analysis for malicious network traffic," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[18] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2022–2030.

[19] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of TLS (without decryption)," *J. Comput. Virol. Hacking Techn.*, vol. 14, pp. 195–211, Aug. 2018.

[20] Y.-C. Chen, Y.-J. Li, A. Tseng, and T. Lin, "Deep learning for malicious flow detection," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–7.

[21] G. Zhao, K. Xu, L. Xu, and B. Wu, "Detecting APT malware infections based on malicious DNS and traffic analysis," *IEEE Access*, vol. 3, pp. 1132–1142, 2015.

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

[23] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement*, C. Dovrolis, Ed., Berlin, Germany: Springer, 2005, pp. 41–54.

[24] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1135–1156, 2nd Quart., 2014.

[25] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Experiments Technol.*, Dec. 2014, pp. 271–282.

[26] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proc. Conf. USENIX Secur. Symp.*, vol. 13, pp. 1–17, Jun. 2004.

[27] *Cicflowmeter*. Accessed: Jun. 2020. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter

[28] W. Xiong and J. C.-M. Lee, "Efficient scene change detection and camera motion annotation for video classification," *Comput. Vis. Image Understand.*, vol. 71, no. 2, pp. 166–181, Aug. 1998.

[29] B. Shahraray, "Scene change detection and content-based sampling of video sequences," in *Proc. Digit. Video Compress., Algorithms Technol.*, Apr. 1995, pp. 2–13.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25. Stateline, NV, USA, Dec. 2012, pp. 1097–1105.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

HONG-YEN CHEN received the B.S. degree in electronics engineering from National Taiwan Normal University, Taipei, Taiwan, in 2019. He is currently pursuing the Ph.D. degree with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei. His research interests include cyber security, machine learning, and deep learning.

TSUNG-NAN LIN (Senior Member, IEEE) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1989, and the M.S. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 1993 and 1996, respectively. He then joined EPSON Research and Development, Inc., San Jose, CA, USA, and EMC Corporation, Hopkinton, MA, USA. Since February 2002, he has been with the Department of Electrical Engineering and the Graduate Institute of Communication Engineering, National Taiwan University. He had been the Director of the Division of Network Management of Computer and Information Networking Center, National Taiwan University, and the Vice President and the Director General of the Cybersecurity Technology Institute, Institute for Information Industry. He is a member of the Phi Tau Phi Scholastic Honor Society.