Encrypted Network Traffic Classification Using a Geometric Learning Model

Ting-Li Huoh Yan Luo
Department of Electrical and Computer Engineering
University of Massachusetts Lowell
Lowell, MA, USA

Tong Zhang
Intel Corporation
Santa Clara, CA, USA

***Abstract*—The increasing volume of encrypted traffic from emerging applications has made conventional traffic classification approaches ineffective and called for novel methods for identifying network flows. Recent machine learning and deep learning based approaches have been proposed yet many of them are severely limited by their feature selection and inherent neural network architecture. As network data by nature are of non-Euclidean distance space and carry abundant chronological relationship, we are inspired to utilize geometric deep learning that simultaneously takes into account packet raw bytes, metadata and packet relations for classifying encrypted network traffic. We validate our proposed graph neural network (GNN) models against the reference methods including convolutional neural networks (CNN) and recurrent neural networks (RNN) quantitatively and demonstrate that the proposed graph neural networks outperform the state of art.**

***Index Terms*—encrypted network traffic analysis, network traffic classification, graph neural networks, deep learning**

I. Introduction

Network traffic classification has played a crucial role in network management, traffic engineering and network security [1]. Yet, as the computer network traffic proliferates, differentiating network flows has been a challenge due to the variety and dynamics of emerging network applications. Furthermore, the increasing volume of encrypted network traffic has made many conventional approaches such as deep packet inspection (DPI) ineffective. Classifying encrypted network flows without decrypting the traffic is of particular importance for privacy reasons.

The usage of machine learning and deep learning models for network traffic classification has been on the rise [2]–[5]. Traditional machine learning methods, such as support vector machine (SVM) and random forest (RF), make use of hand-crafted features that are extracted from raw data. Since the feature dimension is high in network packets and flows, identifying a highly effective set of features is not always a trivial task. Nowadays, data-driven neural methods have achieved a great success in many fields. For instance, recent advances in deep learning, particularly convolution neural networks (CNN) exploit local information by taking all the adjacent points of the center point under the receptive field into account in imaging processing, and long short-term memory (LSTM), one of recurrent neural networks (RNN) members reasons in classification tasks in time series.

978-3-903176-32-4 © 2021 IFIP

Incorporating neural network with the advantage of using raw data [6] has been proved successful to a significant extent. The nature of neural networks is to seek features from the input data by comparing their predictions with the data true labels. In the context of network traffic data, the models will use raw bytes from the packet headers and attempt to learn hidden features from a large amount of network packets with application labels. The commonly used models such as CNN and RNN require uniformly identical data width on every samples due to their network architecture. Therefore, such neural network models cannot inherently differentiate data sample dimensions in training and inferences even though the packet headers differ in length and can carry important application-specific information.

Recently, geometric deep learning has drawn a lot of attention as it aims to generalize neural network models to non-Euclidean domains such as graphs and manifolds. As an example, a graph neural network (GNN) that is able to indicate relationship between nodes in a graph has started to gain its popularity among researchers [7], [8]. We observe that a geometric learning model has the opportunity to represent the hidden features of network data which are naturally in a non-Euclidean domain. We believe that GNN has several salient characteristics that highlight the advantages of graphs for network traffic analysis. Firstly, the arbitrarily structured graphs contribute to the generality of graph network that it could solve problems in non-Euclidean domain. Following from that is the flexibility of network input shape: the shape of each subject can be different during training and validation for GNN, while the consistent input shape is a requisite for both CNN and LSTM. Moreover, another outstanding advantage of graph network architecture is that it can incorporate global attributes, which are shared by all the nodes, assigned for each input graph. In this work, we are motivated to answer the open question: Is a graph neural network able to effectively classify encrypted network traffic flows?

The network traffic analysis at a flow level has been proved to be more accurate and meaningful because there is time series information in flows [9]. In our work, we utilize an open source library [8] to implement a flow-based GNN model with an encoder and decoder structure for modeling encrypted

network traffic classification. Our GNN models are examined on two scenarios. One is that the GNN is trained and validated using a VPN traffic dataset, and the other is using a non-VPN traffic dataset. For each scenario, we also compare our GNN against two reference methods.

The main contributions of our work can be summarized as follows:

- To the best of our knowledge, we are the first to implement a graph neural network architecture on network traffic classification. According to our results, GNN shows its potential to solve problems in differentiating applications in encrypted traffic.
- We present a new notion to translate network traffic flows into graph representations as network inputs that include raw bytes, metadata, as well as chronological relationship altogether.
- We demonstrate the proof-of-concept of the graph neural network for encrypted network traffic classification, and show the feasibility and generality of the proposed approach by conducting experimental studies using both VPN and non-VPN datasets.

The rest of this paper is organized as follows. In Section II, we survey the related works. In Section III, we explain the motivation and rationale behind our research. In Section IV, we describe the datasets as well as the graph network architecture including training procedures. In Section V, we present and discuss the experiment results, and compare the performance of graph network architecture with two reference methods: a CNN model and an LSTM model. Last, in Section VI, we summarize this work and discuss its strengths, limitations and future work.

II. Related Work

Much effort has been devoted to network traffic classification to date for solving all kinds of tasks, such as detecting intrusion and malware from normal network traffic, classifying different applications, and distinguishing network protocols that is applied to specific network packet or flow [10], [11]. As the pervasion of applying techniques such as port forwarding, random ports assignment, network address translation (NAT) and encryption, and the evolution of the awareness of user privacy, conventional network traffic classification methods such as port-based approach and deep packet inspection are considered not to be sufficient to cope with network analysis problems nowadays [12]. Consequently, machine learning-based methods which are reliant on a predefined set of input features have been proposed and widely investigated in network traffic analysis domain. In terms of the machine learning model, several statistical techniques such as Gaussian mixture model (GMM) or support vector machine (SVM) have been applied to solve network traffic classification problems [13]. Also, there are studies which use random forest (RF), C4.5, or k-nearest neighbor (KNN) as algorithms to build the classifiers [14], [15]. However, the main drawback of these feature-based methods is that they rely heavily on expert knowledge for feature engineering.

Recently, models based on deep learning methods, such as CNNs and LSTMs, have achieved a great success in the network traffic classification [16]. Several works have utilized the ISCXVPN2016 dataset [17] to tackle the problem of classifying applications of encrypted network traffic [3], [18]. CNN-based networks are known for the weight-sharing scheme, which uses a window of shared weights, sliding across the input and extracting local information. In [3], Wang et al. proposed an end-to-end encrypted network traffic classification method with 1D-CNN. They took raw bytes of flows as network inputs and showed deep learning methods has its potential on the domain of network traffic analysis. LSTM-based networks are used to process time series data. In [18], Yao et al. implemented an LSTM-based model and took the first ten packets of each flow as network inputs. They also applied attention mechanism to observe the contribution of each packet through the trainable attention vector. Their results show the network tends to focus on the first four packet of a flow since the first few packets carry more protocol-related information. Among similar deep learning-based studies, their proposed networks are mostly trained using only raw data as network inputs for solving the task of network traffic classification. In our work, we view our data as non-Euclidean datasets and develop a geometric learning model, which takes a combination of raw bytes, metadata, and chronological relationship into account, for encrypted network traffic classification.

III. Motivation

Geometric learning has drawn tremendous attention nowadays, as there are numerous existing non-Euclidean datasets in nature such as social networks, transportation networks, and brain networks. Consequently, as examples of geometric learning, graph neural networks that are able to indicate relationship between nodes in a graph have started to gain their popularity among researchers [7], [8]. There are several characteristics that highlight the advantages of graph networks. Firstly, the arbitrarily structured graphs contribute to the generality of graph networks to solve problems in non-Euclidean space. Secondly, it provides the flexibility of network input shape.

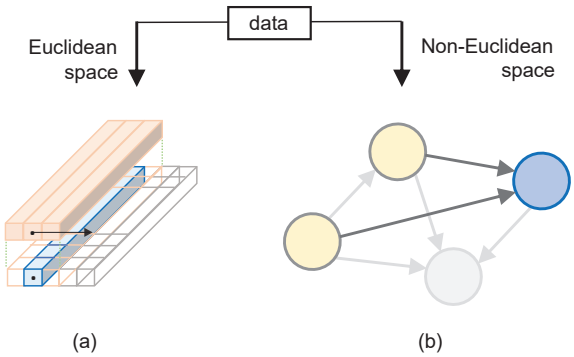

Fig. 1. (a) 1D-CNN: kernel sizes to determine the blue grid's neighbors. (b) GNN: edges to define neighbors for the blue node.

The shape of each sample can be different during training or validation process for GNNs, while the consistent input shape is a requisite for general CNNs and LSTMs. Moreover, another outstanding advantage of graph network architecture is that it can incorporate global attributes, which are shared by all the nodes and assigned for each input graph.

In contrast, it is widely known that CNNs are very productive when solving problems in images which are naturally Euclidean and own the property of translational invariance [19]. As shown in Fig.1, CNNs rely on kernels that determine neighbors for the center grid by their shapes, while GNNs allow researchers to define edges for determining neighbors for each node. In many network traffic studies, researchers map network traffic data into Euclidean domain and then use the Euclidean-based data to train a CNN model [3], [5], [20]. In terms of the network inputs for flow-based network traffic classification, the main drawback of CNN models is that all inputs are limited to a fixed shape once their architectures are determined. For instance, when mapping traffic flows into Euclidean space, the CNN is restricted to take a fixed number of packets for each flow. In the event that a flow has fewer packets, zero padding will be required, which might harm the network performance; when a flow has more packets than the predefined fixed number of packets, the flow will have to be truncated in order to be consistent with the predefined packet number limit, resulting in losing the fidelity of data.

Intuitively, there are several reasons why classifying network traffic data can benefit from GNN models:
(1) Given the fact that GNNs can accept flow data with arbitrary numbers of packets as inputs, we can obviate the needs of the data truncation and zero padding. In other words, each traffic flow can be translated to a graph representation completely without losing important or adding redundant information, which maintains the data fidelity.
(2) With the capability of defining relations between packets in GNNs, network traffic flows can be easily converted into a space which is non-Euclidean and researchers can preserve more original looks and characteristics of flows without Euclidean distance constraints.
(3) Since the graph network architecture allows us to assign global attributes, which are at graph level and are shared by all the nodes, for each input graph, we can easily incorporate additional universal information, such as packet length mean values, and duration of a flow.

These intuitions motivate us to extend network traffic classification task to a geometric deep learning model with graph inputs. The proposed model consists of nodes (raw bytes), global attributes (metadata), as well as edges (chronological relation) altogether. We plan to compare it directly with references deep learning models such as CNNs and RNNs.

IV. Methods

A. Data Description and Preprocessing

We use the UNB ISCX Network Traffic VPN-nonVPN (ISCXVPN2016) dataset [17], which is an encrypted network traffic dataset and contains several types of network traffic and applications. A regular session and a session over VPN are captured using Wireshark and tcpdump, and the encrypted traffic has a total amount of 25 GB of data. In this study, we utilize two subsets of the ISCXVPN2016 dataset for our experimental studies, and it will be denoted as **NonVPN-dataset** and **VPN-dataset** in the remainder of this paper. Under NonVPN-dataset, there are six different types of network traffic across sixteen different applications. As for VPN-dataset, there are six different types of network traffic across fourteen different applications.

In the following, we describe three major steps for data preprocessing, which are data segmentation, masking out biased fields of packets, and adjusting input data sizes and formats according to the graph-structured representations.

1) Data segmentation: SplitCap, an open source tool to split PCAP files based on different criteria, is used to extract traffic flows as the first step. Studies show that using bi-

TABLE I
NonVPN-dataset

Label	Application	No. of flows	Total
Chat	Aimchat	383	2,058
	Facebook	471	
	Gmail	391	
	Hangouts	409	
	ICQ	404	
Email	Email	5,703	5,703
File	FTP	508	1,191
	SCP	157	
	SFTP	181	
	Skype	345	
Streaming	Netflix	270	1,278
	Spotify	144	
	Vimeo	310	
	Youtube	554	
P2P	Torrent	667	667
VoIP	Facebook	535	3,975
	Hangouts	1,869	
	Skype	456	
	Voipbuster	1,115	

TABLE II
VPN-dataset

Label	Application	No. of flows	Total
Chat	Aimchat	30	3,987
	Facebook	1,153	
	Hangouts	2,731	
	ICQ	29	
	Skype	44	
Email	Email	286	286
File	FTPS	42	773
	SFTP	19	
	Skype	712	
Streaming	Netflix	142	525
	Spotify	107	
	Vimeo	117	
	Youtube	159	
P2P	Torrent	262	262
VoIP	Facebook	1,325	6,990
	Hangouts	3,160	
	Skype	896	
	Voipbuster	1,609	

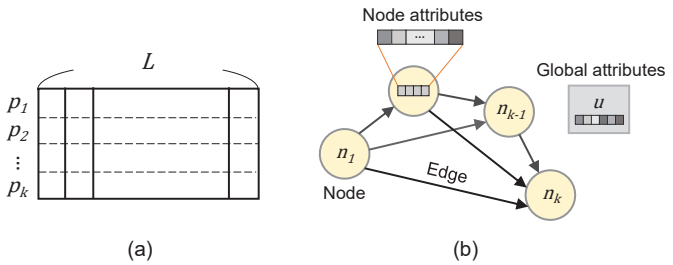

Fig. 2. (a) Shape of a traffic flow, where each row represents a single packet. (b) The graph-structured representation of a flow.

directional flows, also known as sessions, can achieve better performance comparing to unidirectional flows [3]. As a result, for data segmentation, we employ bi-directional flows in our experiments.

The SplitCap application takes a single PCAP file as input and output extracted PCAP files based on 5-tuple. After the extraction, the NonVPN-dataset and VPN-dataset has $14,872$ and $12,823$ bi-directional flows in total, respectively. The extracted flows are split into two parts that 60 percent is for the training dataset and the remaining is for validation. An extracted PCAP file stores a single flow, and each flow consists of packets where their source and destination IP address, source and destination port number, and the protocol in use are the same. Every packet contains a byte stream up to maximum transmission unit (MTU) size of $1,500$ bytes. Table I shows the distribution of flows for each class and application in NonVPN-dataset, and Table II shows the distribution of flows for each class and application in VPN-dataset.

2) Mask out biased fields: For each packet of the extracted flows, the data link layer information is removed, and the source and destination IP addresses are masked. As the original dataset is captured artificially, the source to capture the network traffic is limited to a few hosts or servers. Consequently, the source and destination IP addresses in the IP datagram header are masked with zeros, and the Ethernet header which includes Media Access Control (MAC) addresses information is removed entirely. In this way, the network can refrain from getting biased information as inputs, which may mislead the network during the training process.

3) Network input: The network takes in a flow as an input. As shown in Fig. 2-a, the structure of a flow consists of packets, denoted as $p_{1..k}$, where k is the total number of packets in a flow, and L is our desired lengths of packet bytes. The normalization is performed so that the values for each byte are scale to the range of $[0, 1]$. We evaluate different values of L in different setup of experimental studies listed in Table III. The detailed description of the experiments will be provided in Section IV-B.

In our work, each flow is translated into a graph representation. A graph consists of a set of nodes and edges, and it is also capable of carrying global attributes along with the graph. One key advantage of graph network architecture in terms of network inputs is the flexibility of its input shape. During training and validation, the shape of each subject can be various for GNNs, while the same input shape is a requisite for both CNNs and LSTMs. As shown in Fig. 2-b, the figure shows an example of a graph input. Each packet in the flow is seen as a node, which are denoted as $n_{1..k}$, and each node carries L bytes from the packet. In addition to this, we can assign edges to indicate the relation between nodes. The directed edge from n_{k-1} to n_k, is used to express the chronological relationship of packets among the flow.

Moreover, it is also an advantage of graph network architecture that it can have global attributes assigned for each input graph. In part of our experimental studies, we also include system-level properties of flow that are represented by global attributes, which is denoted as u. In our case, the global attributes comprise seven of the common metadata features extracted from each flow for every network input sample. These flow attributes include source and destination port numbers, payload and packet length mean, standard deviation of payload and packet length, and the duration of flow. Each global attribute is normalized into the range of $[0, 1]$. Consequently, the graph network is able to have a combination of inputs with raw data and metadata training together.

B. Experimental Details

The nature of deep learning methods is to seek hidden features from the given input data by itself. Deep learning methods are not only able to take hand-crafted features as inputs, but also very successful in working with raw data [6]. Hence, we design experiments to assess the impacts of network inputs on GNN, specifically metadata, raw bytes, and chronological relationship. In the following, we describe four studies with different combination of inputs as well as two reference methods to be compared against. The experimental studies are summarized in Table III. In our work, we conduct the experimental studies twice with two datasets, NonVPN-dataset and VPN-dataset, independently, in order to show how the proposed approach generalizes on different datasets.

TABLE III
EXPERIMENTAL STUDIES AND INPUT DATA.
(*For the input types, "Global" represents "metadata", "Node" represents "raw bytes", and "Edge" represents "chronological relationship".*)

Study index	Input types	No. of raw bytes (L)
1	Global + Node + Edge	$1,500$
2	Node + Edge	$1,500$
3	Global	-
4	Global + Node + Edge	100

1) Study 1 and Study 4: In Study 1 and 4, both raw bytes and metadata features are included for network inputs. The raw bytes are mapped into a graph representation with node attributes and edges, that every packet in a flow is a node and each node carries its raw data, and the edges indicate the chronological relationship of packets in a flow. In addition, the metadata features are mapped to the field of global attributes along the network training process. The difference between these two studies is that Study 4 only covers the header

information that the size of packet length was set to 100 bytes. Via Study 1, we would like to examine whether the network can take advantage from the combination of network inputs that the metadata is provided as additional information by making a comparison with Study 2. Moreover, we would like to compare Study 1 with Study 4, in terms of the effectiveness of having payload information included or not for network inputs.

2) Study 2 and Study 3: As for Study 2 and Study 3, both experiments take only either raw data or metadata as network inputs. In Study 2, we train GNN using raw data with a packet length of $1,500$ bytes; while in Study 3, the GNN takes only global attributes into account. We would like to observe and make comparison of the performance of GNN having raw data and metadata as inputs respectively.

3) Reference methods: We choose two commonly used deep learning models, CNN and LSTM, as our reference architectures, which take raw bytes as inputs. Due to the constraint that the number of packets is required to be fixed in order to train CNN and LSTM models, hence, each flow has a shape of $10 \times 1,500$ as in [18], where 10 is the first ten packets of each flow, and $1,500$ is the packet length. Zero padding is applied if a flow has fewer than 10 packets.

C. Network Architecture

The fundamental of our geometric learning model is a graph neural network (GNN) that supports various graph-structured representations. By taking one of the key advantages of its design principle that a graph neural network can be composed of multi-block architectures, we adopt an encode-process-decode architecture for our GNN model. As shown in Fig. 3, our end-to-end graph neural network comprises three functional blocks: a GN_{enc} block, a GN_{core} block, and a GN_{dec} block. For each block, there are three update functions for nodes, edges and global, which are set to 5-layers multilayer perceptron (MLP), where each layer has a total number of 128 neurons, following by a rectified linear unit (ReLU) and a batch norm layer. The aggregation functions used for all the update functions are summation.

To be specific, the GN_{enc} is an encoder that maps the input graph, denoted as x, to a latent domain. The GN_{core} takes the latent space representation, denoted as h, as input to generate the output, denoted as $\hat{h}$, that the GN_{core} is formed by N GN blocks sequentially where $N = 3$. The GN_{dec} then maps the latent space representation, $\hat{h}$, back to generate the final output, which is denoted as $\hat{y}$.

1) Regularization: Regularization is an important pillar to prevent the network from overfitting. Two common approaches are used in this work, penalized weights and dropouts.

The inverse category frequency (ICF) weighting scheme is adopted in our work that a penalized weight is assigned to each class. Since the number of samples of all classes within each batch may be different, the frequencies and weights are being calculated and applied within each batch during network training process. To such a degree, the network will consider the number of samples of all classes to be balanced and contribute equally. Moreover, an important advantage of applying penalized weight according to the class distribution within each batch is that the network would tend to have better adaptability over different class distributions of validation set.

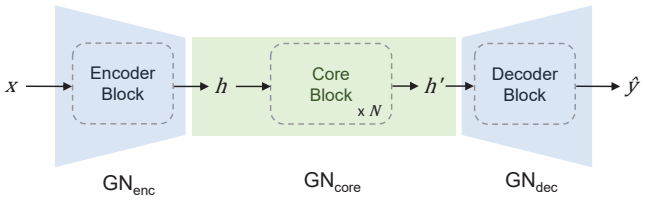

Fig. 3. The graph neural network (GNN) architecture.

Dropout regularization has been considered an efficient regularization method for neural networks [21] that partial of the neurons will be switched off randomly during each training iteration. Both non-recurrent and recurrent dropout techniques are applied to our GNN. Non-recurrent dropout is used in the encoder and decoder blocks, while recurrent dropout is used in the core block.

2) Loss function: The network is trained using cross-entropy as the loss function, which is expressed as:

$$\text{loss}(\hat{y}, y) = \alpha[y] \left(-\hat{y}[y] + \log \left(\sum_i exp\left(\hat{y}[i]\right) \right) \right) \quad (1)$$

where $\hat{y}$ and y are vector prediction results and class true labels respectively, and α is the vector of penalized weights computed from the ICF weighting scheme.

D. Network Training

The graph network is implemented on Tensorflow platform using Graph Nets library created by DeepMind [8] with an NVIDIA V100 graphics card. The graph network takes a graph, consisting of metadata as global attributes, raw bytes as node attributes, and packet relations as edges, as an input and yields a graph with a prediction stored in global. The cross-entropy loss is computed by comparing the prediction with the ground truth. The parameters of the graph network are updated by minimizing the loss using Adam optimizer with the learning rate 0.0003. The batch size is set to 128. With these hyper-parameters, we train our graph network for 500 epochs.

E. Evaluation Metrics

In this work, overall accuracy is not considered a suitable metric because real datasets like network traffic flows, in general, are exceedingly imbalanced, and a high overall accuracy can be easily achieved when a network is trained to favor major categories with a huge number of samples. Therefore, instead of using overall accuracy, the evaluation metrics used to validate our graph neural network are sensitivity and F1 score, which are more emphasizing each category's performance.

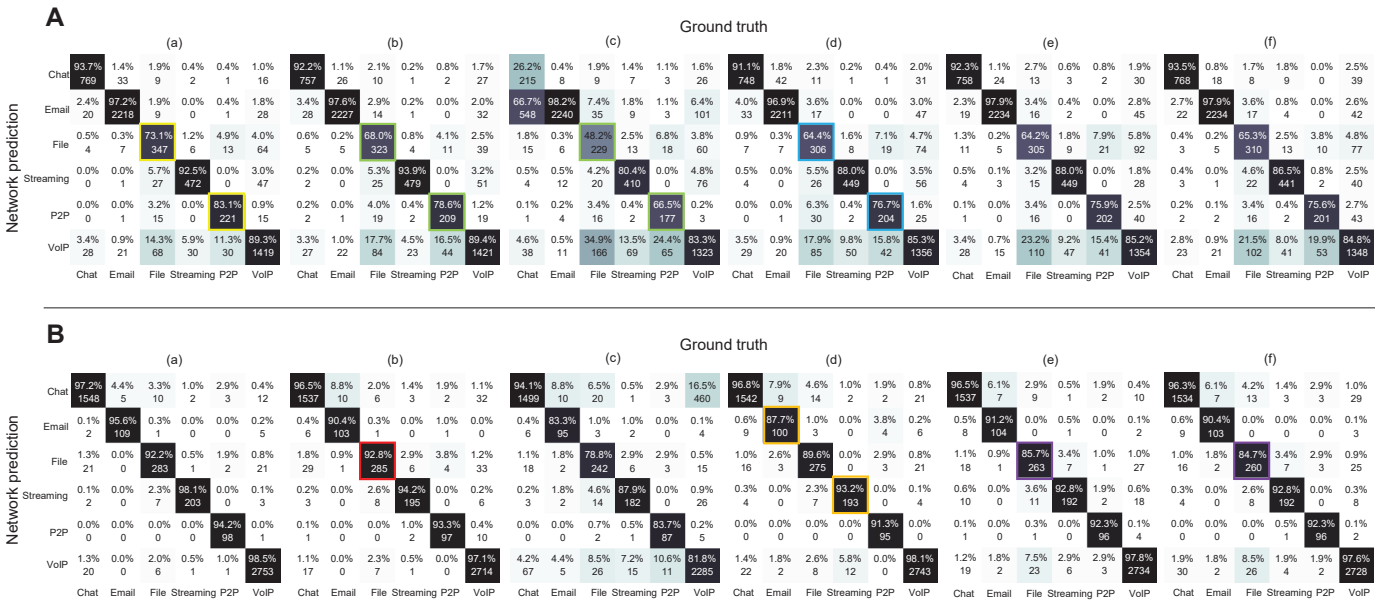

Fig. 4. The confusion matrices across different experimental studies and reference methods with two datasets. Part A presents the result from NonVPN-dataset, and part B presents the result from VPN-dataset. For each square box in a confusion matrix, the lower value is the instance count of correct predictions to ground truth, and the upper value is the normalized number of it. The normalized values have a sum of 1 across columns within a confusion matrix. (a) Study 1 that trained with raw bytes and metadata. (b) Study 2 that trained with raw bytes only. (c) Study 3 that trained with metadata features only. (d) Study 4 that trained with raw bytes and metadata. The raw bytes only covers the header fields. (e) A CNN model that trained with raw bytes. (f) An LSTM model that trained with raw bytes.

1) Sensitivity: The sensitivity is also called recall. It measures, within samples which have the same label, the ratio of number of the correctly predicted samples to the total number of the samples and is defined as:

$$\text{Sensitivity}_i = \frac{TP_i}{TP_i + FN_i}, \tag{2}$$

where TP is the number of true positive samples, FN is the number of false negative samples, and i indicates the class index.

2) Precision: The precision measures, within samples which are classified as the same label, the proportion of number of the correctly predicted samples to the total number of the samples and is defined as:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}, \tag{3}$$

where FP is the number of false positive samples.

3) F1 score: The F1 score, which computes the harmonic mean of the sensitivity and the precision, is a widely used metric for models trained with an imbalanced dataset and is defined as:

$$\text{F1}_i = 2 \times \frac{\text{Sensitivity}_i \times \text{Precision}_i}{\text{Sensitivity}_i + \text{Precision}_i}. \tag{4}$$

V. Results

A. NonVPN-dataset

Fig. 4A showcases the confusion matrices where we can see how much a network predicts correctly for each class across our experimental studies and reference methods for NonVPN-dataset. The confusion matrix of Study 1 is shown in Fig. 4A-a, that the network is trained with a combination of full raw data and metadata as inputs. By comparing to the green boxes in Study 2 (Fig. 4A-b) and Study 3 (Fig. 4A-c), the yellow boxes from Study 1 shows the sensitivities of File and P2P are boosted by about 5 percent. A key observation here is that when the global attributes are incorporated with the node attributes and the edges along the training process, it helps achieve better performance of the multi-class classifier. It implies that the graph network takes advantages of using the additional input of metadata, which requires extra knowledge to acquire and networks cannot easily decode and learn properly from raw data during training.

The graph networks of Study 2 and Study 3 are trained with full raw bytes only and metadata only, respectively. According to the corresponding confusion matrices, Fig. 4A-b and Fig. 4A-c, it is obvious that the graph neural network trained with raw data outperforms that trained with metadata in terms of the overall performance. The result is consistent with the prior research [6] that neural network models are more suitable to be trained with raw data, and have been proved successful to a significant extent.

The confusion matrix of Study 4 is shown in Fig. 4A-d, that the network is trained with inputs consisting of raw bytes and metadata, similar to Study 1, yet covering only the header information rather than the full-length bytes. By comparing the blue boxes in Study 4 (Fig. 4A-d) with the yellow boxes in Study 1 (Fig. 4A-a), there is a decrease of 8.7 and 6.4 percent in sensitivities of File and P2P respectively. According to the observation, it implies that apart from the header information, the payload field may also contain important information so

TABLE IV
NONVPN-DATASET: PERFORMANCE COMPARISON

NonVPN-dataset	Network input	Metric	Chat	Email	File	Streaming	P2P	VoIP
Study 1	Raw + Metadata ($L = 1,500$)	Sensitivity	**0.936**	0.972	**0.730**	0.925	**0.830**	0.893
		Precision	**0.926**	**0.974**	0.786	0.862	**0.877**	**0.889**
		F1 score	**0.931**	**0.973**	**0.757**	0.893	**0.853**	**0.891**
Study 2	Raw only ($L = 1,500$)	Sensitivity	0.922	0.976	0.680	**0.939**	0.785	**0.894**
		Precision	0.919	0.967	**0.834**	0.860	0.829	0.876
		F1 score	0.920	0.971	0.749	**0.897**	0.806	0.885
Study 3	Metadata only	Sensitivity	0.261	**0.982**	0.482	0.803	0.665	0.832
		Precision	0.802	0.762	0.671	0.785	0.871	0.791
		F1 score	0.394	0.858	0.561	0.794	0.754	0.811
Study 4	Raw + Metadata ($L = 100$)	Sensitivity	0.911	0.969	0.644	0.880	0.766	0.853
		Precision	0.896	0.958	0.726	0.839	0.778	0.857
		F1 score	0.903	0.963	0.683	0.859	0.772	0.855
CNN	Raw only ($L = 1,500$)	Sensitivity	0.923	0.979	0.642	0.880	0.759	0.852
		Precision	0.913	0.964	0.688	**0.899**	0.779	0.848
		F1 score	0.918	0.971	0.664	0.890	0.769	0.850
LSTM	Raw only ($L = 1,500$)	Sensitivity	0.935	0.979	0.652	0.864	0.755	0.848
		Precision	0.912	0.963	0.741	0.866	0.755	0.848
		F1 score	0.923	0.971	0.694	0.865	0.755	0.848

TABLE V
VPN-DATASET: PERFORMANCE COMPARISON

VPN-dataset	Network input	Metric	Chat	Email	File	Streaming	P2P	VoIP
Study 1	Raw + Metadata ($L = 1,500$)	Sensitivity	**0.971**	**0.956**	0.921	**0.980**	**0.942**	**0.985**
		Precision	0.979	**0.931**	0.862	**0.944**	0.989	0.989
		F1 score	**0.975**	**0.943**	**0.891**	**0.962**	**0.965**	**0.987**
Study 2	Raw only ($L = 1,500$)	Sensitivity	0.964	0.903	**0.928**	0.942	0.932	0.971
		Precision	0.966	0.927	0.796	0.919	0.881	**0.990**
		F1 score	0.965	0.915	0.857	0.930	0.906	0.980
Study 3	Metadata only	Sensitivity	0.941	0.833	0.788	0.879	0.836	0.817
		Precision	0.752	0.863	0.846	0.801	0.915	0.948
		F1 score	0.836	0.848	0.816	0.838	0.874	0.878
Study 4	Raw + Metadata ($L = 100$)	Sensitivity	0.968	0.877	0.895	0.932	0.913	0.981
		Precision	0.969	0.819	**0.864**	0.927	**1.000**	0.984
		F1 score	0.968	0.847	0.880	0.930	0.954	0.982
CNN	Raw only ($L = 1,500$)	Sensitivity	0.964	0.912	0.856	0.927	0.923	0.978
		Precision	**0.981**	0.904	0.829	0.824	0.941	0.981
		F1 score	0.973	0.908	0.842	0.872	0.932	0.979
LSTM	Raw only ($L = 1,500$)	Sensitivity	0.963	0.903	0.846	0.927	0.923	0.976
		Precision	0.965	0.895	0.830	0.905	0.969	0.977
		F1 score	0.964	0.899	0.838	0.916	0.945	0.976

that the model's performance can be more accurate and reliable when taking full raw data as network inputs.

Fig. 4A-b, Fig. 4A-e and Fig. 4A-f are confusion matrices for Study 2, CNN, and LSTM models, where raw data is served as network inputs. According to the quantitative results, we confirm that the graph neural network is comparable to, and even slightly better than CNN and LSTM. It is likely that because the graph network can obtain extra information provided by the edges that are used to indicate the chronological relationship between packets within a flow.

The performance metrics used for evaluating our experimental studies and reference methods for NonVPN-dataset are tabulated in Table IV. The table indicates training with full raw data and metadata together as network inputs performs better than other experimental studies as well as the reference methods, CNN and LSTM models. In terms of the sensitivity, half of the classes (Chat, File and P2P) in Study 1 achieve the highest values among all the experimental studies and the two reference methods, and in particular, Study 1 model's performance on File and P2P even exceeds all other methods by a wide margin. As for the precision, four out of six classes in Study 1 achieve the highest values among all the other experiments listed. Since F1 score computes the harmonic mean of the sensitivity and the precision, as expected, Study 1 has the highest F1 values in most of the categories.

B. VPN-dataset

In our work, we went through the above mentioned experimental setup once again with our second dataset, VPN-dataset. Fig. 4B showcases the confusion matrices across our experimental studies and the reference methods for VPN-dataset, and the performance metrics used for validating our work are tabulated in Table V. Consistent with the result obtained from NonVPN-dataset, the GNN model performs the best when metadata is incorporated with raw data along the training process. As shown in 4B-a, all the categories, except for File in Study 1, have the highest sensitivities, comparing with Study 2 (Fig. 4B-b) and Study 3 (Fig. 4B-c). Also, in terms of F1 score, all categories of Study 1 achieve the highest values among all the other experiments listed. According to the result, we conclude that GNN can take advantage

from the additional input of metadata, which requires extra knowledge to acquire and networks cannot easily decode and learn properly from raw data during training.

Similar trend occurs that some of the categories' accuracies will drop by a wide margin when the network input only covers header information. By comparing with Study 1 (Fig. 4B-a), the orange boxes from Study 4 (Fig. 4B-d) shows the accuracies of Email and Streaming have a decrease of 7.9 and 4.9 percent respectively, and the accuracies of rest of the classes are also lower than Study 1. Hence, through the observation, we conclude that the model's performance could be more accurate and reliable when taking full raw data as network inputs, rather than only providing the header fields.

Fig. 4B-b, Fig. 4B-e, and Fig. 4B-f are confusion matrices of Study 2, CNN, and LSTM, respectively, which all take raw bytes as inputs. Likewise, since GNN can obtain extra information provided by the edges that are used to indicate the chronological relationship between packets, our GNN model is comparable to, and even slightly better than our two reference methods according to the quantitative results. In particular, the sensitivity of File category as indicated in the red box from Study 2 (Fig. 4B-b) substantially exceeds the sensitivities of two reference methods, in the purple boxes on Fig. 4B-e and Fig. 4B-f, by approximately 8 percent.

Quantitatively, according to the showcased confusion matrices in Fig. 4B and the performance metrics in Table V that are used for evaluating and validating our experimental studies and reference methods for VPN-dataset, we conclude that GNN trained with full raw data and metadata together as inputs outperforms the remaining experimental studies as well as the reference CNN and LSTM models.

VI. Conclusion

In this work, we implement a flow-based geometric learning model that simultaneously takes raw bytes, metadata and chronological relationship into account for classifying encrypted network traffic. We compare our models against the reference methods and demonstrate that: for File and P2P in NonVPN-dataset and for File in VPN-dataset, both of the Study 2 models have achieved a great performance comparable to the CNN model's and the LSTM model's, and the Study 1 models substantially outperform Study 2. These results show that incorporating both raw bytes and metadata into a model is advantageous. In the future work, as a strategy to validate our model's more robust performance, we plan to acquire more encrypted network traffic data with more category labels to increase our training size.

Acknowledgment

This work is supported in part by Intel Corporation and NSF No. 1738965.

References

[1] S. Bagui, X. Fang, E. Kalaimannan, S. C. Bagui, and J. Sheehan, "Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features," *Journal of Cyber Security Technology*, vol. 1, no. 2, pp. 108–126, 2017.

[2] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*. IEEE, 2017, pp. 712–717.

[3] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 43–48.

[4] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[5] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1271–1276.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[8] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," 2018.

[9] G. Marín, P. Casas, and G. Capdehourat, "Deep in the Dark - Deep Learning-Based Malware Traffic Detection Without Expert Knowledge," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 36–42.

[10] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo, "A Survey on Encrypted Traffic Classification," in *International Conference on Applications and Techniques in Information Security*. Springer, 2014, pp. 73–81.

[11] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[12] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012.

[13] M. Dusi, A. Este, F. Gringoli, and L. Salgarelli, "Using GMM and SVM-Based Techniques for the Classification of SSH-Encrypted Traffic," in *2009 IEEE International Conference on Communications*. IEEE, 2009, pp. 1–6.

[14] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic using Time based Features." in *ICISSP*, 2017, pp. 253–262.

[15] K. Shahbar and A. N. Zincir-Heywood, "How far can we push flow analysis to identify encrypted anonymity network traffic?" in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–6.

[16] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[17] 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 *ICISSP*, 2016.

[18] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of Encrypted Traffic Through Attention Mechanism Based Long Short Term Memory," *IEEE Transactions on Big Data*, pp. 1–1, 2019.

[19] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A Guide to Convolutional Neural Networks for Computer Vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, 2018.

[20] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.