Flow-based Encrypted Network Traffic Classification with Graph Neural Networks

Ting-Li Huoh, Yan Luo, Peilong Li, and Tong Zhang

***Abstract*—Classifying encrypted traffic from emerging applications is important but challenging as many conventional traffic classification approaches are ineffective, thus calling for novel methods for identifying encrypted network flows. Recent machine learning and deep learning-based approaches are severely limited by their feature selection and inherent neural network architecture. More importantly, they overlook the opportunity to capture latent information in the temporal dimension of packets. As network data by nature are of non-Euclidean distance space and carry abundant chronological and temporal relations, 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. Our proposed graph neural network (GNN) model outperforms the two reference methods, convolutional neural networks (CNN) and recurrent neural networks (RNN) quantitatively as indicated by three metrics: sensitivity, precision and F1 score.**

***Index Terms*—Encrypted network traffic analysis, network traffic classification, deep learning, graph neural networks, multimodal deep learning.**

I. Introduction

IDENTIFYING network traffic is a vital task to enable the functionalities of governing systems in network management and network security. Network traffic classification has been intensively explored in light of current improvements in computer networking. While ongoing research covers a wide range of topics, such as malware detection, intrusion detection, and application prediction, the common goal is to accurately and effectively differentiate network traffic, which is evidently difficult due to the dynamics and complexity of emerging network applications. Additionally, when the volume of encrypted network data grows, many conventional techniques, such as deep packet inspection (DPI) [1], become inefficient, and it is particularly crucial for privacy concerns to classify encrypted network flows without decrypting the traffic.

Extensive research has been conducted on network traffic classification, as classifying network traffic is critical for managing system functionality in network management and security. In the early days, conventional network traffic classification methods, such as port-based approaches and DPI, mainly focused on packet-level analysis. Port-based approaches use merely the port information to identify packets, and therefore quickly become obsolete due to networking technology advancements, such as dynamic ports assignment and network address translation (NAT) [2]. On the other hand, DPI inspects the payload information to classify network packets [1], yet this method violates user privacy to a significant extent. Meanwhile, with the ever-increasing security level and volume of Internet services, applying DPI to encrypted network traffic analysis also becomes ineffective [2].

Subsequently, researchers begin to adopt machine learning approaches in network traffic analysis. Machine learning approaches rely heavily on predefined input features and require domain knowledge for feature engineering, which is the process of extracting hand-crafted features from raw data. As a result, much effort has been expended on discovering critical flow statistical features at the flow-level, rather than at the packet-level, and evaluating their effectiveness using various machine learning-based classifiers [3]–[6], such as support vector machine (SVM) and random forest (RF) on network traffic classification.

With the rapid development of neural networks, deep learning-based methods have become state-of-the-art across a wide range of applications, including network traffic analysis [7]. For flow-level network traffic classification tasks, convolutional neural networks (CNN) and recurrent neural networks (RNN) models perform exceptionally well in predicting 1D or 2D Euclidean space datasets. However, the disadvantage is that when data is mapped into the Euclidean spaces, valuable latent information derived from packet relations in the flow data is lost. Hence, we introduce a concept for mapping network traffic flows into non-Euclidean graph representations with packet relations that preserves data integrity to the greatest extent possible, as well as a methodology for classifying network traffic in the non-Euclidean domains.

We utilize graph format as input to our graph neural networks (GNN) architecture. Then we design an end-to-end flow-based GNN model with an encoder and decoder structure to classify both function types and application types of encrypted network traffic flows, and train our GNN model using a dataset with VPN encryption protocols. We leverage the DeepMind Graph Nets library [8], which is a library for building graph neural networks in TensorFlow and Sonnet, to build our GNN model, and the network architecture design details are presented in Section III-C. In order to show the viability and applicability of the proposed graph-based approach, apart from training the GNN model using encrypted dataset to classify network function types and application types of encrypted flows, we also train our GNN model with a hybrid of encrypted and non-encrypted dataset.

Accordingly, the main contributions of our work can be summarized as follows:

- We leverage a novel geometric learning framework [8] and propose a multimodal graph-based deep learning approach for flow-based encrypted network traffic classification, which can effectively differentiate application

TABLE I
NOTABLE RELATED WORKS USING DEEP LEARNING IN NETWORK TRAFFIC CLASSIFICATION TASKS

Reference	Deep Learning Classifier	Multimodal Learning	Dataset		Network Input	Raw Data	Flow	
			public	private			uni-	bi-
This paper	GNN	✔	✔	-	raw bytes $(1,500\text{B})$ + 6 flow features/statistics + packet relations	✔	-	✔
Shen et al., 2021 [16]	GNN	-	-	✔	packet length (25 packets) + client-server interactions	-	-	✔
Pang et al., 2021 [17]	GNN	✔	✔	✔	raw bytes $(1,500\text{B})$ + adjacency relationship	✔	-	✔
Huoh et al., 2021 [18]	GNN	✔	✔	-	raw bytes $(1,500\text{B})$ + 8 flow features/statistics + chronological relationship	✔	-	✔
Aceto et al., 2021 [14]	1D-CNN + BiGRU	✔	✔	-	L4 app. layer (784B) + 4 flow features (32 packets)	✔	-	✔
Shapira & Shavitt, 2021 [19]	2D-CNN	-	✔	-	packet size + inter-packet time (60-second block)	-	✔	-
Zhang et al., 2020 [11]	2D-CNN, LSTM	-	✔	✔	packet-based raw bytes $(1,521\text{B})$ / flow-based statistics	✔	-	✔
Aceto et al., 2019 [13]	1D-CNN + BiGRU	✔	✔	✔	L4 app. layer (516B) + 4 flow features (12 packets)	✔	-	✔
Yao et al., 2019 [10]	Attention + LSTM	-	✔	-	L4 app. layer $(10\times1,500\text{B})$ / raw bytes $(10\times1,500\text{B})$	✔	-	✔
Zeng et al., 2019 [12]	1D-CNN + LSTM + SAE	-	✔	-	raw bytes (900B)	✔	-	✔
Chen et al., 2017 [20]	2D-CNN	-	-	✔	packet size + direction + inter-packet time (10 packets)	-	-	✔
Lopez-Martin et al., 2017 [21]	2D-CNN + LSTM	-	-	✔	6 flow features (20 packets)	-	-	✔
Wang et al., 2017 [9]	1D-CNN	-	✔	-	L4 app. layer (784B) / raw bytes (784B)	✔	✔	✔

and function types in encrypted traffic. Among existing studies on network traffic prediction tasks, the state-of-the-art methods train deep learning-based networks with raw bytes in the Euclidean domains [9]–[14], whereas our proposed GNN model accepts arbitrary graph inputs that can embody data in the non-Euclidean domains, preserving data integrity to the greatest extent possible.

- In terms of network inputs for our GNN model, we present a notion to map network traffic flows into graph representations. Apart from taking raw bytes as inputs, a merit of our work is that we further enhance our GNN by supplementing the second modality of data, where meta features are served as graph-level attributes, and packet relations are served as additional variables that include temporal information and chronological relationship.
- Furthermore, we experimentally compare the performance with different combinations of network inputs to validate our proposed approach using a public dataset, ISCXVPN2016 [15], and demonstrate the GNN's proof-of-concept for classifying network traffic flows according to their function and application types, and establish a foundation for future graph-based studies on network traffic flow classification problems.

The rest of this article is organized as follows. In Section II, we survey the evolution of approaches and literature in network traffic classification, and explain the rationale behind our research. In Section III, we describe and demonstrate the proposed approach using graph neural networks for encrypted network traffic classification. Our experimental results and main findings are reported in Section IV. In Section V, we discuss the strengths and limitations, and present future directions of our work. Last, we summarize our work in Section VI.

II. Related Work

A. Deep Learning for Network Traffic Classification

Because of the fast growth of neural networks, deep learning-based approaches have become state-of-the-art across a broad variety of applications, including network traffic analysis [7]. Unlike machine learning-based methods, deep learning-based methods, namely neural networks, which are capable of discerning important hidden features and representations from their input data, can be applied directly to learn raw data without any extra feature extraction steps. Related works that use deep learning approaches in network traffic classification are summarized in Table I. As shown in Table I, recent works in network traffic classification with deep learning-based methods have explored different types of neural network architectures [9]–[14], [19]–[21], such as convolutional neural networks and long short-term memory (LSTM). CNNs rely on kernels and perform convolution operations to exploit local information under the receptive field, while LSTMs are one of recurrent neural network members that are capable of learning long-term dependencies and temporal information. Experiments from existing work [9]–[11], [20], [21] have demonstrated consistently better classification results with deep learning-based methods than with conventional or machine learning-based methods. 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 [22]. In many network traffic studies, researchers map network traffic data into the Euclidean domains and then use the Euclidean-based data to train a CNN model [9], [11], [23]. 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 the Euclidean spaces, the CNN is restricted to take a fixed number of packets for each flow.

For studying encrypted network traffic classification using deep learning-based methods, ISCXVPN2016 [15], a well-known public network traffic dataset that contains numerous encrypted traces, has been widely used in many research studies [9]–[12], [14], [19]. For instance, using the same network traffic dataset, Wang et al. [9] presented a 1D-CNN model to predict encrypted network traffic, Yao et al. [10] proposed an LSTM-based model with attention mechanism, and Zeng et al. [12] proposed an encrypted network traffic classifier utilizing both CNNs and LSTMs. According to their quantitative results, the studies reported that employing bi-directional flows with all raw bytes as network inputs leads to better performance. Nevertheless, there are studies use merely flow features or flow-based statistical information as network inputs [11], [19]–[21]. Subsequently, a multimodal deep learning approach has been proposed for network traffic classification tasks. Aceto et al. [13] reported that multimodal approaches are able to leverage network traffic data according to various types of network inputs, where the multimodal deep learning classifier could capitalize the heterogeneity of both intra- and inter-modality dependencies of network traffic data [14].

B. Graph Neural Networks

Recently, geometric deep learning has drawn a lot of attention as it aims to generalize neural network models to the non-Euclidean domains, such as social networks, transportation networks, and brain networks. While widely-used deep learning architectures, such as CNNs and RNNs, are capable of exploiting hidden features of data in the Euclidean domains, they cannot handle non-Euclidean data that is represented as graphs. As a result, a few studies have been devoted to encrypted network traffic classification employing graph neural networks, and are listed in Table I. The studies have shown the approach that leverages the information-rich graph representation of network traffic flows is competitive to other common deep learning-based methods [16]–[18]. Our previous work has translated network traffic flows into graph representations and demonstrated the proof-of-concept of the graph neural network for encrypted network application classification [18]. Likewise, in [17], Pang et al. also presented a chained graph representation to capture the causal relationship for the raw network traffic data. Different from our previous work [18], Pang et al. [17] only considered packet raw bytes and the adjacency relationship between packets within a network traffic flow, whereas we simultaneously take packet raw bytes,

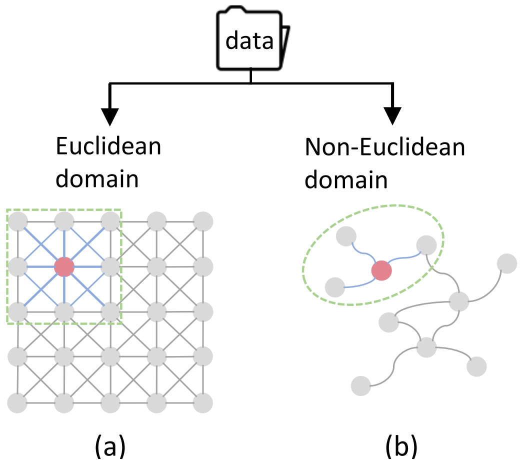


Fig. 1. (a) 2D-CNN: kernel sizes to determine the red grid's neighbors. (b) GNN: edges to define neighbors for the red node.

meta features and chronological relationship between packets into account. Shen et al. [16] focused on decentralized applications (DApps) identification using graph neural networks, and in their work, the client-server interactions in terms of packet lengths and directions within a network flow are encapsulated in the traffic interaction graph (TIG).

In terms of network architecture, there are three major differences between graph networks and common deep learning architectures. First, GNNs can be directly applied to graphs, meaning that the geometric deep learning methods can deal with data in non-Euclidean space, which can preserve more original looks and characteristics of data. Secondly, CNN and RNN models, due to their network structure constraints, require zero padding or data trimming to keep input data shape identical during training and validation process. Unlike CNN or RNN models, GNNs can obviate this limitation and allow arbitrary input shapes for each subject within a dataset. Last, for each input graph, additional graph-level attributes can be assigned as a universal information for each graph, while CNN and RNN models can only accept elements with the same data type for an input.

The basic computation unit in the Graph Networks framework is a graph network (GN) block [8], which takes a graph as an input, performs message passing through update and aggregation functions, and yields a graph as an output. An input graph is composed of a set of nodes and edges with graph-level attributes. The nodes describe the vertices of a graph, and the directed edges indicate the connections between nodes. Properties associated with nodes and edges are defined as node attributes and edge attributes respectively. As shown in Fig. 1a, when a kernel from a CNN slides over a grid-structured input to examine matching features, the center of the kernel can determine all the resulting convolved neighbors. As a contrast in Fig. 1b, we can use the edges within non-Euclidean graph structures to determine the neighbors of each nodes so that GNNs can use the information in the neighborhood of a node to compute the updated value.

Our previous work on network traffic classification utilize deep learning-based approach [18], and this work is an exten-

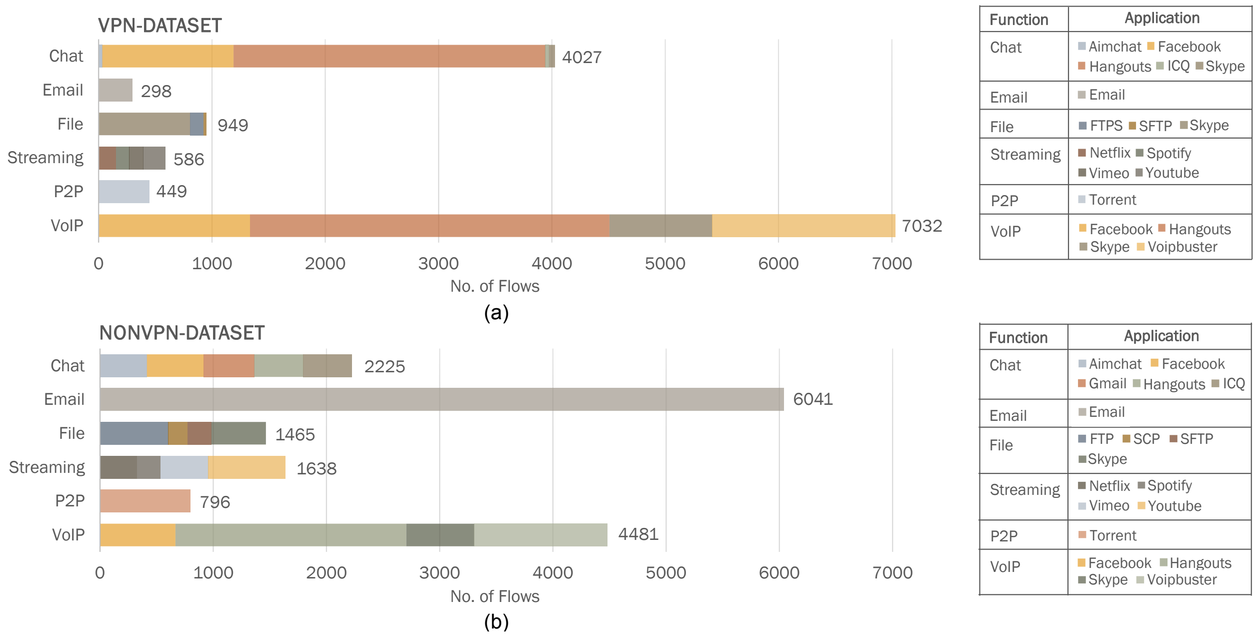


Fig. 2. Distribution of flow diagrams by function type in two datasets: (a) VPN-dataset, and (b) NonVPN-dataset.

sion of the effort. With the aforementioned salient properties of graphs and GNNs, in [18], we presented a notion to translate network traffic flows into graph representations as network inputs that include raw bytes, meta features, and chronological relationship between packets. Additionally, we extended network traffic classification task to a geometric learning model using graph neural networks, and showed GNN has the potential to solve problems in differentiating function types of encrypted network traffic flows.

In accordance with the discourse in Section II, the highlights of the additional effort compared to our earlier work are summarized in the following. In this work, a public dataset, ISCXVPN2016 [15], is employed to validate our GNN model. Port numbers are removed to prevent biased conclusions or misleading classification outcomes as stated in [14], and more experiments are conducted to demonstrate the viability and applicability of the proposed graph-based approach. In addition to classifying function types, experiments for classifying application types of network traffic flows are also provided. Furthermore, investigations are also conducted in which the GNN model is trained using both encrypted and non-encrypted traces. Additionally, we use multimodal deep learning by supplying raw bytes and flow features into our GNN model as network inputs. Instead of merely encapsulating the chronological relationship into edges [18], the temporal information, which is the inter-arrival time between packets within a network traffic flow, is collected as an additional network input to the GNN, as inter-arrival time is considered as valuable information for network traffic classification [6]. As such, we demonstrate the mapping of a network traffic flow to a graph-structured representation in which each packet is assigned to a node, packet relations are encapsulated in edges with the chronological relationship serving as the edge direction and temporal information serving as the edge weight, and meta features of a flow are assigned to global attributes. The detail of the complete graph inputs will be presented in Section III-B.

III. Methods

In this section, we present the detailed implementation of the end-to-end geometric deep learning model for encrypted network traffic classification. The contents include introducing the experiment dataset, presenting how the data is preprocessed and translated into graph representations for GNN, describing the experimental design and details, explaining our GNN architecture and the procedures of network training and validation, and the evaluation metrics being used.

A. Data Description

In this study, we select the UNB ISCX Network Traffic VPN-nonVPN (ISCXVPN2016) dataset [15] for our experiments. It is a public dataset with virtual private network (VPN) encryption protocols that has been widely adopted in the research area of network traffic classification recently [9]–[11], [14]. A VPN is a secured network that allows for secure data transmission, and a VPN encryption protocol is a process used to establish a securely encrypted path between two endpoints. With VPN service, network users' data will pass through an encrypted VPN tunnel to reach the VPN server that acts as a gateway to the public Internet. The original ISCXVPN2016 dataset has a total amount of 25 GB of data stored in the network packet capture (PCAP) format, which contains a regular session and a session over VPN that are captured using Wireshark and Tcpdump. Application-wise, the dataset contains several function types of network traces such as email, file transfer, and streaming traces; and encryption-wise, the dataset consists of VPN traces and non-VPN ones. In this study, we will use the two subsets of

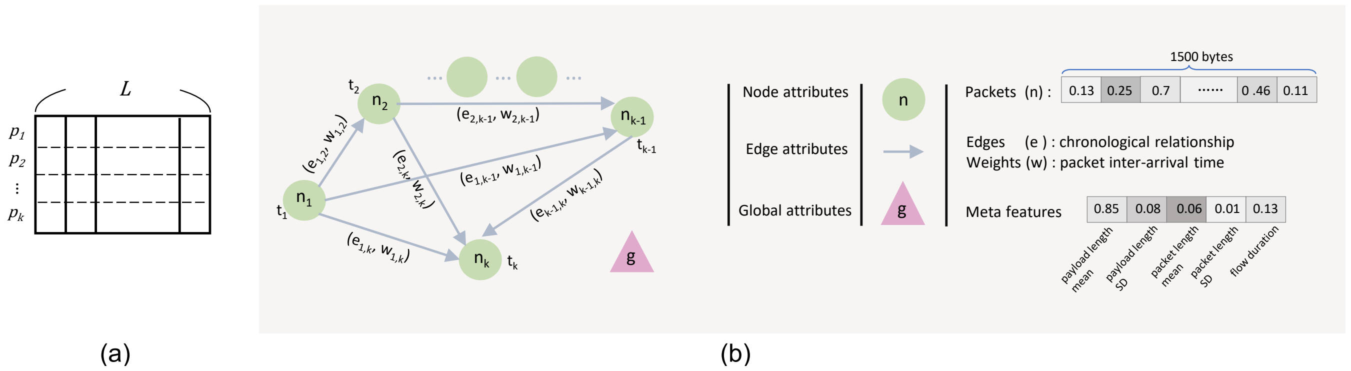


Fig. 3. (a) The architecture of a network flow. (b) A graph-structured representation of a flow's overall structure, as well as detailed information on the node, edge, and global attributes.

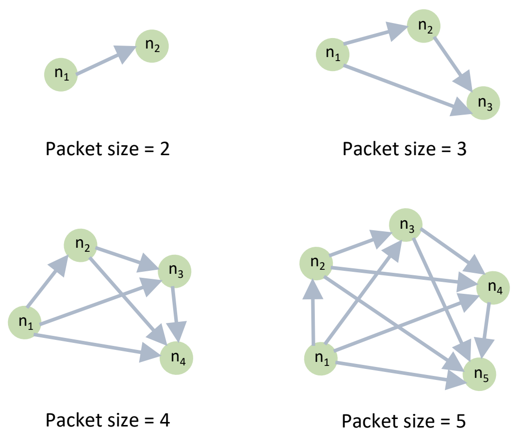


Fig. 4. Instances of graph inputs with different flow packet sizes, ranging from 2 to 5.

the ISCXVPN2016 dataset; the subset of VPN traces will be referred to as **VPN-dataset**, and the subset of non-VPN traces will be referred to as **NonVPN-dataset**. Within the VPN-dataset, there are six network traffic types across fourteen different applications. As for the NonVPN-dataset, there are six different types of network traffic across sixteen different applications. A complete list of the included network function types and application types for VPN-dataset and NonVPN-dataset can be found in Fig. 2a and Fig. 2b, respectively.

B. Network Input

Since the GNN takes in a graph as an input, it is imperative to translate network traffic data into graph-structured representations, which consist of three building blocks: a set of nodes (Node) and edges (Edge) with graph-level attributes (Global). In the following, we describe data preprocessing steps which includes data segmentation and masking out biased fields of packets, as well as how we translate a network flow into a graph as network input.

1) Data Preprocessing: One of the prior works [9] shows that leveraging bi-directional flows, also known as sessions, help achieve better performance comparing to unidirectional flows, as a result, we employ bi-directional flows in our work. To extract flow-based data, data segmentation technique is performed to obtain flow samples from the original PCAP files. A bi-directional flow, also known as session, can be identified by the criteria of 5-tuple where the source and destination can be transposed. The 5-tuple consists of source IP address, source port, destination IP address, destination port, and transport protocol. After the extraction, VPN-dataset and NonVPN-dataset have $13,341$ and $16,646$ bi-directional flows in total, respectively. For each bi-directional flow, it consists of a sequence of packets where every packet contains a byte stream up to maximum transmission unit (MTU) size of $1,500$ bytes, and prior research [10] shows that most of the flows consist of fewer than 5 packets in the ISCXVPN2016 dataset. Fig. 2a shows the distribution of flows for each function type and application type in VPN-dataset, and Fig. 2b shows the distribution of flows for each function type and application type in NonVPN-dataset. For both VPN-dataset and NonVPN-dataset, the extracted flows are split into two parts that 60% is for the training dataset and the remaining is for validation.

As the original dataset was captured manually, it is possible that the collected network traces have common hosts and servers. Hence, to avoid carrying biased information which might favor the deep learning model during the training process, for every packet, both the Ethernet header that contains Media Access Control (MAC) addresses and the source and destination IP addresses that are embedded in the IP datagram header are removed. Also, port numbers are removed to avoid biased conclusions or inflated classification outcomes as mentioned in [14]. After the above-mentioned data preprocessing steps, as shown in the Fig. 3a, 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. Each bytes' values are normalized such that they fall inside the interval $[0, 1]$. Rather than only covering the header fields, in this work, full raw data are taken as network inputs, where L is set to $1,500$, as the quantitative results in prior works [9], [10], [18] have shown that the model's performance can be more accurate and reliable when taking full raw data as network inputs.

2) Graph Representation Generation: Fig. 3b illustrates a general graph-structured representation. Each flow is transformed to a graph representation in our work. A graph is composed of a collection of nodes and edges, and it is also capable of containing global properties. The details of how a network traffic flow is mapped to each graph entity is presented in the following.

a) Node: Within a network traffic flow, each packet is mapped to a node, denoted as n_i, where $i = \{1..k\}$, and k is the total number of packets or nodes. In each node, it stores the raw bytes as node attributes, where each byte is normalized to $[0, 1]$. The dimension of the node attributes is set to $1,500$, which corresponds to the MTU size.

b) Edge: Edges can be assigned to show the relationship between nodes. In this work, the edges are used to indicate the chronological relationship and the temporal information between packets. As Fig 3b shows, chronological relationship between packets is viewed as directed edges, denoted as $e_{s,r}$. Packet's timestamps are denoted as t_i, where $i = \{1..k\}$, and k is the total number of packets or nodes. Temporal information, which is the inter-arrival time between packets and can be calculated by $t_r - t_s$, is stored as the weight of the edge, denoted as $w_{s,r}$. In this scenario, s is the sender node index and r is the receiver node index. For instance, as Fig. 3b shows, the time difference between packet n_{k-1} and packet n_k are assigned to edge attributes, denoted as $w_{(k-1),k}$ while the direction of edge from n_{k-1} to n_k, denoted as $e_{(k-1),k}$, are used to express the chronological relationship between packets. The packet timestamp is used to allocate edges. Each node is connected to the nodes whose packets arrive earlier through arriving traversing edges and to the nodes whose packets arrive later via leaving traversing edges. Fig. 4 depicts, with different flow packet sizes ranging from 2 to 5, how nodes are connected using edges. In the message passing algorithm, each node is updated by aggregating messages provided from surrounding nodes through arriving traversing edges. We design our GNN to consider all previous packets/nodes while processing the current packet/node, and those former packets/nodes can directly deliver messages to the current node, boosting message passing speed.

c) Global: Lastly, meta features of a flow are stored in the field of global attributes, denoted as g, which are shared across a graph. In [5], the most important flow features of ISCXVPN2016 dataset are reported, hence, we select and assign five common meta features of a flow to global attributes accordingly, which includes payload length mean, standard deviation of payload length, packet length mean, standard deviation of packet length, and duration of flow. These meta features can be obtained through an open source network analysis tool, Tcpdump. The meta features are normalized to $[0, 1]$ before assigning to global attributes.

As such, we can incorporate raw bytes, packet relations, and metadata as graph network inputs by mapping network traffic flow into a graph-structured representation.

C. End-to-end GNN Architecture

The fundamental of our geometric learning model is a graph neural network that supports various graph-structured representations. As shown in Fig. 5a, the end-to-end GNN is constructed using encode-process-decode scheme [8] and is comprised of an encoder block (GN_{enc}), a string of N core blocks (GN_{core}), where N is set to 5 in this work, and a decoder block (GN_{dec}). To be specific, the GN_{enc} maps the input graph, denoted as G, to a latent domain. The GN_{core} takes the latent space representation, denoted as G_{latent}, as input to generate the output, denoted as G'_{latent}. The GN_{dec} then maps the latent space representation, G'_{latent}, back to generate the final output, which is denoted as G'.

The GN block's structure contains three 5-layer multilayer perceptron (MLP) networks as shown in Fig. 5b, where each layer contains 128 neurons and is followed by a ReLU activation function except for the output layer, as update functions for Edge, Node, and Global. In addition, all the MLP networks are followed by a batch norm layer except for the decoder block's, which stabilizes the network training process. As shown in Fig. 5c, the relation of the three update functions in the encoder and the decoder blocks are independent. The three update functions will construct a latent graph from the input graph and extract information from a latent graph respectively. On the other hand, as shown in Fig. 5d, the three update functions in the core block are linked for message-passing, which is a process of sending information between nodes via the edges.

Regularization is a crucial component of the network training stage, as it prevents networks from overfitting and favoring certain majority classes as a result of data imbalance. We deploy two common approaches, penalized weights and dropouts, in the GNN architecture. For a multi-class classification task, the network performance is usually vulnerable to a data imbalance issue. Thus, we incorporate the inverse category frequency (ICF) weighting scheme in the loss function. Furthermore, every time a batch of data is gathered, the numbers of samples of each class may be different in the batch. Therefore, the frequencies and weights are calculated and applied within each batch during network training process, so that samples of each class can be learned equally by the network. In addition, we consider our GNN trained with the batch-level ICF will tend to have better adaptability when tested on various validation sets with different class distributions. For neural networks, dropout regularization is an efficient regularization method [24], in which partial of the neurons will be switched off randomly during each training iteration so that it can reduce chance of overfitting and help networks learn more hidden features.

All the aggregation functions used for this GNN architecture are summation. The loss function selected for training our GNN is cross-entropy loss function, which is expressed as:

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

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.

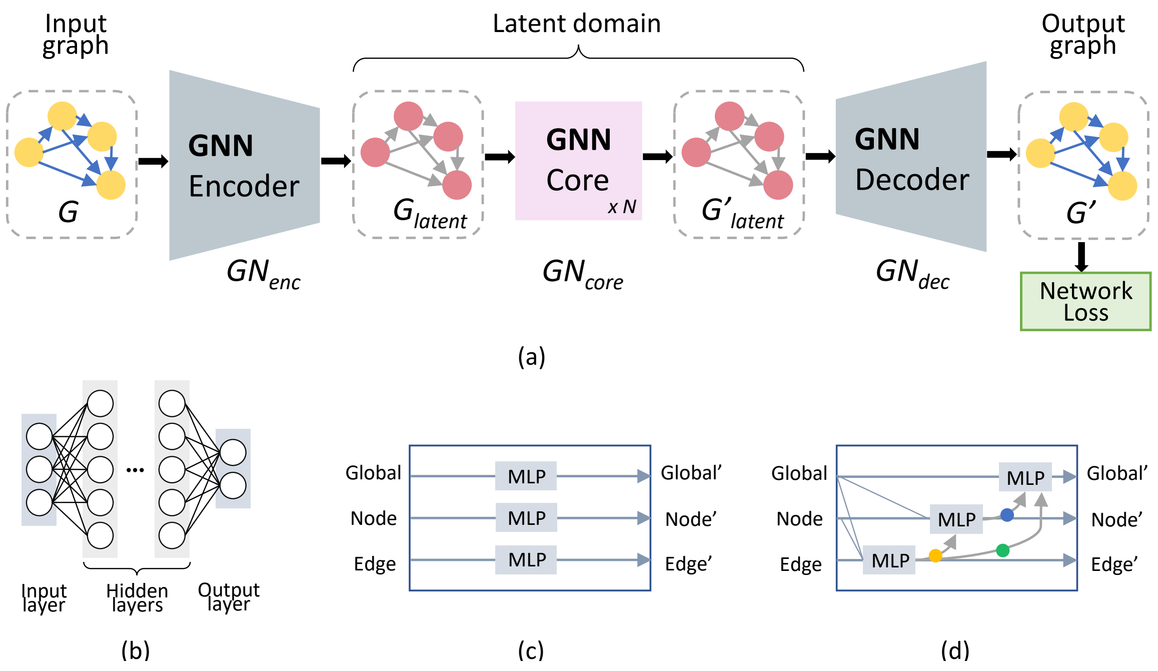


Fig. 5. (a) The proposed end-to-end GNN architecture for network traffic classification. (b) MLP network architecture. (c) The GNN encoder and decoder architecture. (d) The GNN Core architecture (the yellow circle represents a node-level edge aggregation function, the blue circle represents a global-level node aggregation function, and the green represents a global-level edge aggregation function).

D. Experimental Design

In the following, we describe nine experimental studies and two deep learning-based comparison methods, a CNN [9] and an LSTM [10] model. Experiments using various datasets, classification tasks, and graph input element combinations are aimed to evaluate our proposed approach. The classification of function types and application types will be represented by the terms **Function-type** and **Application-type**, respectively. The experimental studies are summarized in Table II. To be specific, Study 1 to Study 6 are GNN models trained using VPN-dataset for Function-type classification. The first six studies are intended to evaluate the effects of network inputs on GNNs, and hence each study uses an unique combination of graph input components, including Node, Global, and Edge. Similarly, to assess the generality of GNNs across classification tasks, Study 7 is a GNN model trained on the VPN-dataset but specifically for Application-type classification. Studies 8 and 9 are intended to study circumstances in which network traces obtained at a particular endpoint may comprise encrypted or non-encrypted flows from various sources. Hence, Study 8 and Study 9 are trained using both VPN-dataset and NonVPN-dataset, and are trained for Function-type classification and Application-type classification, respectively. As for the two reference methods, same as Study 1, both the CNN and the LSTM models take only raw bytes as network inputs and are trained for Function-type classification of network traffic flows using VPN-dataset.

1) Study 1: In Study 1, the GNN takes only Node as a network input. We train GNN using raw data with a packet length of $1,500$ bytes. The GNN model is trained for Function-type classification of network traffic flows using VPN-dataset. Via Study 1, we would like to observe and make comparison of the performance of our GNN model and the two deep learning-

TABLE II
EXPERIMENTAL STUDIES AND INPUT DATA.

Study Index	Dataset	Task	Input Types
1	VPN-dataset	Function-type	Node (raw bytes)
2	VPN-dataset	Function-type	Global (meta features)
3	VPN-dataset	Function-type	Edge (packet relations)
4	VPN-dataset	Function-type	Node (raw bytes) + Global (meta features)
5	VPN-dataset	Function-type	Node (raw bytes) + Edge (packet relations)
6	VPN-dataset	Function-type	Node (raw bytes) + Edge (packet relations) + Global (meta features)
7	VPN-dataset	Application-type	Node (raw bytes) + Edge (packet relations) + Global (meta features)
8	VPN-dataset + NonVPN-dataset	Function-type	Node (raw bytes) + Edge (packet relations) + Global (meta features)
9	VPN-dataset + NonVPN-dataset	Application-type	Node (raw bytes) + Edge (packet relations) + Global (meta features)
CNN [9]	VPN-dataset	Function-type	$10 \times 1,500$B (raw bytes)
LSTM [10]	VPN-dataset	Function-type	$10 \times 1,500$B (raw bytes)

based baseline methods, CNN and LSTM, which also take only raw bytes as network inputs.

2) Study 2: In Study 2, we would like to evaluate the performance of our GNN model, which takes only Global as network inputs. The GNN model is trained for Function-type classification of network traffic flows using VPN-dataset. Moreover, by making a comparison to Study 4 that takes meta features and raw bytes as network inputs, we can observe how meta features supplements the GNN model.

3) Study 3: In Study 3, the GNN takes only Edge as network inputs, and the GNN model is trained for Function-type classification of network traffic flows using VPN-dataset. We would like to evaluate the performance of the GNN model when it takes only Edge as network inputs, and also make a comparison to Study 5 to see how Edge element supplements the GNN model.

4) Study 4: As for Study 4, the input graph to our GNN model consists of both Node and Global. The GNN model is trained for Function-type classification of network traffic flows using VPN-dataset. This study helps to compare with Study 1 and the two deep learning-based baseline methods, which both take only raw bytes as their network inputs. We would like to investigate whether GNN can benefit from the additional meta features that belongs to Global elements.

5) Study 5: In Study 5, the input graph to our GNN model consists of both Node and Edge, and the GNN model is trained for Function-type classification of network traffic flows using VPN-dataset. Similar to Study 4, the underlying reason to conduct this study is to make comparisons between studies that are trained with raw bytes only, such as Study 1 and the two reference methods. Via Study 5, we would like to evaluate whether GNN can benefit from the Edge element, where the temporal information and the chronological relationship of the network traffic data are preserved.

6) Study 6: The GNN in Study 6 takes a graph that includes Node, Edge, and Global as network inputs. The GNN model is trained for Function-type classification of network traffic flows using VPN-dataset. In this case, the GNN will simultaneously take a combination of network inputs into account, which include raw bytes, packet relations, and meta features. With Study 6, we would like to observe if GNN can gain benefit from a graph input when additional meta features and edge information are also included and preserved in addition to raw bytes.

7) Study 7: In Study 7, the GNN takes a graph which includes Node, Edge, and Global as network inputs, and is trained using VPN-dataset. However, different from the aforementioned experimental studies, the GNN model in Study 7 is trained for Application-type classification rather than Function-type classification of network traffic flows. Six common categories of applications in VPN-dataset with sufficient samples are chosen to include in the classification task. The selected applications include Facebook, Hangouts, Skype, Email, Torrent, and Voipbuster. Via Study 7, we would like to evaluate whether the proposed GNN model can generalize on different classification task.

8) Study 8: In Study 8, the GNN also takes a graph which includes Node, Edge, and Global as network inputs, and is trained for Function-type classification of network traffic flows as Study 6. In most cases, the network traces being received at a certain endpoint may contain encrypted or non-encrypted flows from different sources. Hence, different from Study 6, the GNN model in Study 8 is trained using both VPN-dataset and NonVPN-dataset. The GNN model is assessed by testing on the validation sets of VPN-dataset and NonVPN-dataset separately, and also jointly. We would like to examine whether the GNN model that trained with both encrypted and non-encrypted traces can have the ability to classify Function-type of network traffic flows.

9) Study 9: The GNN model in Study 9 has the same network input types as Study 6, Study 7, and Study 8, and is trained for Application-type classification as Study 7. Similar to Study 8, the GNN model is trained using both VPN-dataset and NonVPN-dataset. Six common categories of applications across VPN-dataset and NonVPN-dataset with sufficient samples are chosen to include in the classification task. Same categories of applications being used from Study 7 are selected, which includes Facebook, Hangouts, Skype, Email, Torrent, and Voipbuster. The GNN model is assessed by testing on the validation sets of VPN-dataset and NonVPN-dataset separately, and also jointly. Via Study 9, we would like to examine whether the GNN model that trained with both encrypted and non-encrypted traces can classify Application-type of network traffic flows effectively. We also like to show how the proposed approach can generalize on different classification task through the study.

10) Deep Learning-based Reference Methods: Our baseline approaches are based on two state-of-the-art deep learning architectures, an 1D-CNN by Wang et al. [9] and an attention-based LSTM network by Yao et al. [10]. As in Study 1, both the CNN and the LSTM models take only raw bytes as network inputs and are trained for Function-type classification of network traffic flows using VPN-dataset. In their methods, while the LSTM model uses the first $1,500$ bytes of the first 10 packets within a bi-directional flow, which is the optimal heuristic configuration in [10], as the network input, the CNN model uses only the first 784 bytes of each bi-directional flow as the network input. In its original, unaltered form, the CNN model performs poorly in our initial experiments. To make it more competitive and ensure a fair comparison, we make two major changes to the CNN model: (1) we utilize the first $1,500$ bytes of the first 10 packets inside a flow; and (2) we add 8 more convolutional layers to accommodate more hidden features. For each input sample, the surplus packets are trimmed when a flow has more than 10 packets, and zero padding is applied when a flow has fewer than 10 packets. Also, we employ the same data preprocessing steps as in our work, and the biased information has been removed to avoid biased conclusions or inflated classification outcomes as mentioned in Section III-B1.

E. Model Training and Validation

The graph network is implemented on Tensorflow 2.0 using the DeepMind Graph Nets library [8] and trained with an NVIDIA V100 graphics card. The extracted network traffic flows of VPN-dataset are split into two portions that 60% is for the training dataset, and the remaining is for validation. The graph network accepts an input graph that contains meta features in the form of global attributes, raw bytes in the form of node attributes, and packet relations in the form of edges, and generates and returns a graph with a prediction of Function-type or Application-type stored in Global. The parameters of the graph network are updated by minimizing the network loss using Adam optimizer with the learning rate 0.0003. The ICF cross-entropy loss is computed by comparing the prediction with the ground truth labels, and the learning rate is a hyper-parameter that controls the pace of updating neural networks. The batch size is set to 128, where it is the sample size from the training dataset that will be used to calculate a loss for updating the weights once. With these hyper-parameters, we train our graph network for 300 epochs.

F. Evaluation Metrics

The performance metrics used for evaluating our proposed GNN model are sensitivity, precision, F1 score, and overall accuracy. While the overall accuracy is the most commonly used benchmark for evaluating a model, it may not be considered the only and the most important metric, as real-world datasets are normally imbalanced, and a high overall accuracy can be achieved easily when a network is trained to favor the major categories that have more samples. Therefore, micro metrics are also provided to show prominence to each category's performance, and additionally, the full confusion matrices across our experimental studies and the two reference methods are also reported. Through the given confusion matrices, we can comprehend how much a network predicts correctly for each label in terms of network robustness.

1) Overall Accuracy: The overall accuracy can be computed as the total number of samples that are correctly classified by the network model divided by the total number of samples and is defined as:

$$\text{Overall accuracy} = \frac{\sum_i TP_i}{\sum_i (TP_i + FN_i)}, \tag{2}$$

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

2) Sensitivity: The sensitivity is also known as recall, and 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:

$$\text{Sensitivity}_i = \frac{TP_i}{TP_i + FN_i}, \tag{3}$$

3) 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:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}, \tag{4}$$

where FP is the number of false positive samples.

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

$$\text{F1}_i = 2 \times \frac{\text{Sensitivity}_i \times \text{Precision}_i}{\text{Sensitivity}_i + \text{Precision}_i}. \tag{5}$$

IV. Results

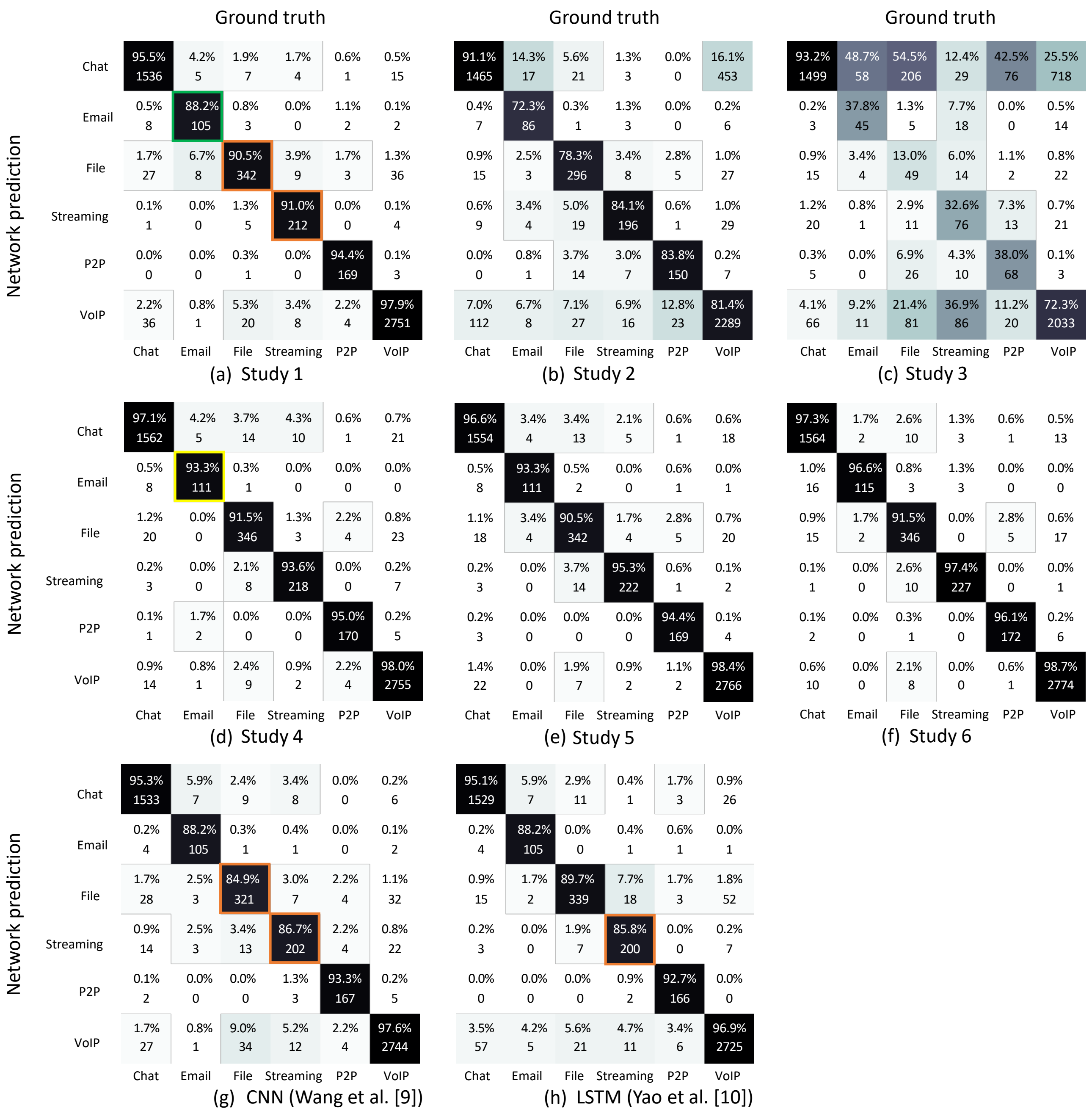
A. VPN-dataset Function-type Classification Results:

The performance metrics used for evaluating our proposed GNN models are tabulated in Table III. Fig. 6 shows a comparison of the confusion matrices for Studies 1 to 6 and the two reference methods. In terms of sensitivity, precision, and F1 score, Study 1 is similar to, and even slightly better than, the two reference approaches, the CNN [9] and the LSTM [10] models, when only raw bytes are used. When the confusion matrix pairs (Study 1, CNN) are compared, it is clear that Study 1 significantly improves the performance of File and Streaming, as indicated by the orange boxes, and when the confusion matrix pair (Study 1, LSTM) is compared, it shows that Study 1 substantially improves the performance of Streaming, as indicated by the orange box. One of the underlying reasons for the GNNs' ability to take flow data with an arbitrary number of packets as network inputs is that the GNNs can give network input shape flexibility, allowing the form of each sample to be different throughout the training and validation processes. When mapping traffic flows into graph representations, all packets in a flow may be kept without redundancy, which benefits the network. The CNN or the LSTM model's inputs are restricted to a fixed number of packets for each flow once the architectures are determined. Zero padding is needed when a flow has fewer packets, which may harm network performance; when a flow has more packets than the predefined shape, the flow will be truncated resulting in losing the fidelity of data.

Study 1 is trained with raw bytes, whereas only meta features are used in Study 2. Comparisons of the confusion matrix pairs (Study 1, Study 2) show that Study 1 outperforms Study 2 significantly in every category, and the result is consistent with the prior research [25] that claims neural network models are appropriate to be trained with raw data because the nature of deep learning methods is to seek hidden features from the given inputs by itself.

The confusion matrix pair (Study 1, Study 4) shows that Study 4 has better performance almost in every category based on the number of correct predictions, and the yellow box in Study 4 indicates that the sensitivity of Email increases by approximately 5.1% comparing to the green box in Study 1. In terms of the performance measures presented in Table III, Study 4 shows an increase in the overall sensitivity, precision, and F1 score, indicating that Study 4 outperforms Study 1. The results indicate that the GNN can benefit from the additional input of meta features, which are difficult to decoded or learn from raw data since they require domain knowledge.

The GNN model in Study 3 takes only the Edge element as network inputs, and the confusion matrix of Study 3 is shown in Fig. 6c. As expected, the number of correct predictions of



(a) Study 1

	Chat	Email	File	Streaming	P2P	VoIP
Chat	95.5% 1536	4.2% 5	1.9% 7	1.7% 4	0.6% 1	0.5% 15
Email	0.5% 8	88.2% 105	0.8% 3	0.0% 0	1.1% 2	0.1% 2
File	1.7% 27	6.7% 8	90.5% 342	3.9% 9	1.7% 3	1.3% 36
Streaming	0.1% 1	0.0% 0	1.3% 5	91.0% 212	0.0% 0	0.1% 4
P2P	0.0% 0	0.0% 0	0.3% 1	0.0% 0	94.4% 169	0.1% 3
VoIP	2.2% 36	0.8% 1	5.3% 20	3.4% 8	2.2% 4	97.9% 2751

(b) Study 2

	Chat	Email	File	Streaming	P2P	VoIP
Chat	91.1% 1465	14.3% 17	5.6% 21	1.3% 3	0.0% 0	16.1% 453
Email	0.4% 7	72.3% 86	0.3% 1	1.3% 3	0.0% 0	0.2% 6
File	0.9% 15	2.5% 3	78.3% 296	3.4% 8	2.8% 5	1.0% 27
Streaming	0.6% 9	3.4% 4	5.0% 19	84.1% 196	0.6% 1	1.0% 29
P2P	0.0% 0	0.8% 1	3.7% 14	3.0% 7	83.8% 150	0.2% 7
VoIP	7.0% 112	6.7% 8	7.1% 27	6.9% 16	12.8% 23	81.4% 2289

(c) Study 3

	Chat	Email	File	Streaming	P2P	VoIP
Chat	93.2% 1499	48.7% 58	54.5% 206	12.4% 29	42.5% 76	25.5% 718
Email	0.2% 3	37.8% 45	1.3% 5	7.7% 18	0.0% 0	0.5% 14
File	0.9% 15	3.4% 4	13.0% 49	6.0% 14	1.1% 2	0.8% 22
Streaming	1.2% 20	0.8% 1	2.9% 11	32.6% 76	7.3% 13	0.7% 21
P2P	0.3% 5	0.0% 0	6.9% 26	4.3% 10	38.0% 68	0.1% 3
VoIP	4.1% 66	9.2% 11	21.4% 81	36.9% 86	11.2% 20	72.3% 2033

(d) Study 4

	Chat	Email	File	Streaming	P2P	VoIP
Chat	97.1% 1562	4.2% 5	3.7% 14	4.3% 10	0.6% 1	0.7% 21
Email	0.5% 8	93.3% 111	0.3% 1	0.0% 0	0.0% 0	0.0% 0
File	1.2% 20	0.0% 0	91.5% 346	1.3% 3	2.2% 4	0.8% 23
Streaming	0.2% 3	0.0% 0	2.1% 8	93.6% 218	0.0% 0	0.2% 7
P2P	0.1% 1	1.7% 2	0.0% 0	0.0% 0	95.0% 170	0.2% 5
VoIP	0.9% 14	0.8% 1	2.4% 9	0.9% 2	2.2% 4	98.0% 2755

(e) Study 5

	Chat	Email	File	Streaming	P2P	VoIP
Chat	96.6% 1554	3.4% 4	3.4% 13	2.1% 5	0.6% 1	0.6% 18
Email	0.5% 8	93.3% 111	0.5% 2	0.0% 0	0.6% 1	0.0% 1
File	1.1% 18	3.4% 4	90.5% 342	1.7% 4	2.8% 5	0.7% 20
Streaming	0.2% 3	0.0% 0	3.7% 14	95.3% 222	0.6% 1	0.1% 2
P2P	0.2% 3	0.0% 0	0.0% 0	0.0% 0	94.4% 169	0.1% 4
VoIP	1.4% 22	0.0% 0	1.9% 7	0.9% 2	1.1% 2	98.4% 2766

(f) Study 6

	Chat	Email	File	Streaming	P2P	VoIP
Chat	97.3% 1564	1.7% 2	2.6% 10	1.3% 3	0.6% 1	0.5% 13
Email	1.0% 16	96.6% 115	0.8% 3	1.3% 3	0.0% 0	0.0% 0
File	0.9% 15	1.7% 2	91.5% 346	0.0% 0	2.8% 5	0.6% 17
Streaming	0.1% 1	0.0% 0	2.6% 10	97.4% 227	0.0% 0	0.0% 1
P2P	0.1% 2	0.0% 0	0.3% 1	0.0% 0	96.1% 172	0.2% 6
VoIP	0.6% 10	0.0% 0	2.1% 8	0.0% 0	0.6% 1	98.7% 2774

(g) CNN (Wang et al. [9])

	Chat	Email	File	Streaming	P2P	VoIP
Chat	95.3% 1533	5.9% 7	2.4% 9	3.4% 8	0.0% 0	0.2% 6
Email	0.2% 4	88.2% 105	0.3% 1	0.4% 1	0.0% 0	0.1% 2
File	1.7% 28	2.5% 3	84.9% 321	3.0% 7	2.2% 4	1.1% 32
Streaming	0.9% 14	2.5% 3	3.4% 13	86.7% 202	2.2% 4	0.8% 22
P2P	0.1% 2	0.0% 0	0.0% 0	1.3% 3	93.3% 167	0.2% 5
VoIP	1.7% 27	0.8% 1	9.0% 34	5.2% 12	2.2% 4	97.6% 2744

(h) LSTM (Yao et al. [10])

	Chat	Email	File	Streaming	P2P	VoIP
Chat	95.1% 1529	5.9% 7	2.9% 11	0.4% 1	1.7% 3	0.9% 26
Email	0.2% 4	88.2% 105	0.0% 0	0.4% 1	0.6% 1	0.0% 1
File	0.9% 15	1.7% 2	89.7% 339	7.7% 18	1.7% 3	1.8% 52
Streaming	0.2% 3	0.0% 0	1.9% 7	85.8% 200	0.0% 0	0.2% 7
P2P	0.0% 0	0.0% 0	0.0% 0	0.9% 2	92.7% 166	0.0% 0
VoIP	3.5% 57	4.2% 5	5.6% 21	4.7% 11	3.4% 6	96.9% 2725

Fig. 6. VPN-dataset Function-type classification. Confusion matrices for eight studies: (a) Study 1: the GNN input is a graph consisting of Node (raw bytes) only; (b) Study 2: the GNN input is a graph consisting of Global (meta features) only; (c) Study 3: the GNN input is a graph consisting of Edge (time information) only; (d) Study 4: the GNN input is a graph consisting of Node (raw bytes) and Global (meta features); (e) Study 5: the GNN input is a graph consisting of Node (raw bytes) and Edge (time information) ; (f) Study 6: the GNN input is a graph consisting of Node (Raw bytes), Edge (time information), and Global (meta features); (g) The CNN model: the input is raw bytes; (h) The LSTM model: the input is raw bytes.

each categories are significantly lower than other experimental studies because the network takes merely the information of packet relations without knowing the raw bytes or meta features as network input. However, as shown in the confusion matrix derived from Study 5 (Fig. 6e), where the GNN model is trained using raw bytes with time information supplemented, the sensitivities of each categories are higher than or equal to Study 1 that is trained using raw bytes only. According to the quantitative results, the provided temporal information and chronological relationship between packets within a flow has the efficacy to boost the performance of the GNN model.

Furthermore, according to Table III, Study 6, our proposed approach, which considers meta features, raw bytes, and packet relations as GNN inputs at the same time, outperforms all GNN variants and the two reference methods in terms of the overall accuracy, has the highest sensitivity values in every category, and four out of the six categories have the highest precision and F1 score values. The evaluation result indicates that the GNN model's performance is improved by obtaining additional information from the Global, which

TABLE III
PERFORMANCE COMPARISON

VPN-dataset	Network Input	Overall Accuracy	Metric	Chat	Email	File	Streaming	P2P	VoIP	Overall
Study 1	Raw bytes	96.00%	Sensitivity	0.955	0.882	0.904	0.909	0.944	0.978	92.92%
			Precision	0.979	0.875	0.804	0.955	0.976	0.975	92.78%
			F1 score	0.967	0.878	0.851	0.931	**0.960**	0.977	92.78%
Study 2	Meta features	84.12%	Sensitivity	0.911	0.722	0.783	0.841	0.838	0.814	81.84%
			Precision	0.747	0.835	0.836	0.759	0.838	0.924	82.36%
			F1 score	0.821	0.774	0.808	0.798	0.838	0.866	81.79%
Study 3	Packet relations	70.76%	Sensitivity	0.932	0.378	0.129	0.326	0.379	0.723	47.82%
			Precision	0.579	0.529	0.462	0.535	0.607	0.885	59.98%
			F1 score	0.714	0.441	0.202	0.405	0.467	0.796	50.45%
Study 4	Raw bytes + Meta features	96.88%	Sensitivity	0.971	0.932	**0.915**	0.935	0.949	0.980	94.75%
			Precision	0.968	0.925	0.873	0.923	0.955	0.989	**93.92%**
			F1 score	0.969	**0.928**	0.894	0.929	0.952	0.984	94.32%
Study 5	Raw bytes + Packet relations	96.92%	Sensitivity	0.966	0.932	0.904	0.952	0.944	0.984	94.75%
			Precision	0.974	0.902	0.870	0.917	0.960	0.988	93.55%
			F1 score	0.970	0.917	0.887	0.934	0.952	0.986	94.13%
Study 6	Raw bytes + Meta features + Packet relations	**97.56%**	Sensitivity	**0.972**	**0.966**	**0.915**	**0.974**	**0.960**	**0.986**	**96.27%**
			Precision	**0.981**	0.839	**0.898**	**0.949**	0.950	**0.993**	93.55%
			F1 score	**0.977**	0.898	**0.906**	**0.961**	0.955	**0.990**	**94.83%**
CNN [9]	Raw bytes	95.20%	Sensitivity	0.953	0.882	0.849	0.867	0.933	0.976	91.02%
			Precision	0.980	0.929	0.812	0.782	0.943	0.972	90.36%
			F1 score	0.966	0.905	0.830	0.822	0.938	0.974	90.63%
LSTM [10]	Raw bytes	95.05%	Sensitivity	0.950	0.882	0.896	0.858	0.927	0.969	91.42%
			Precision	0.969	**0.937**	0.790	0.921	**0.988**	0.964	92.86%
			F1 score	0.960	0.909	0.840	0.888	0.956	0.967	92.03%

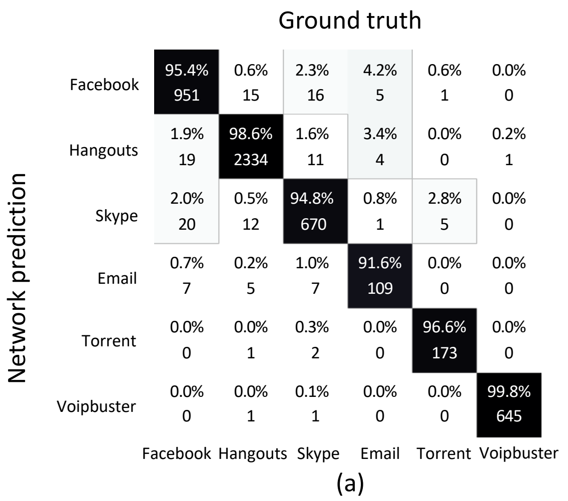


Fig. 7. VPN-dataset Application-type classification. A confusion matrix for Study 7.

can include system-level properties of flow represented by global attributes, and the Edge, which is used to indicate the chronological relationship and temporal information.

In addition, we conducted an additional experiment in which the GNN model is trained without the ICF weighting scheme for Study 6 The sensitivities for each category are as follows: Chat: 0.972, Email: 0.907, File: 0.925, Streaming: 0.875, P2P: 0.949, and VoIP: 0.988. By comparing the experiment without ICF to the one with ICF (Fig. 6f), the sensitivities of the three categories with minor samples, Email, Streaming, and P2P, have decreased by 5.9%, 9.9%, and 1.1%, respectively. One important finding is that the GNN model without ICF is relatively incompetent to recognize and learn samples from the minor categories, whereas the GNN model with ICF (Study 6) tends to learn samples from each class equally.

B. VPN-dataset Application-type Classification Results:

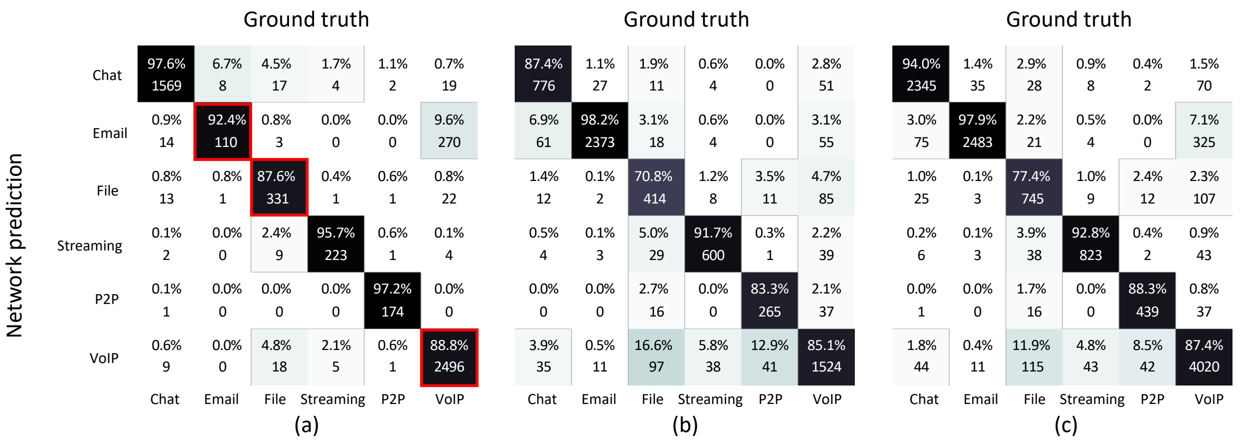
In Section IV-A, we demonstrate several experiments with different combination of graph input elements, and assess the impacts of network inputs on GNN. We also make comparisons of the performance of our GNN model and the two deep learning-based baseline methods, the CNN and the LSTM, and show our proposed approach that incorporating raw bytes, metadata, and packet relations into the GNN model is advantageous. We apply the proposed approach to a different classification task in this subsection to see whether the proposed GNN model can generalize to other classification tasks. Hence, Study 7 is conducted and trained for Application-type classification rather than Function-type classification of network traffic flows using VPN-dataset. The performance metrics used for validating Study 7 are tabulated in Table IV, and the confusion matrix of Study 7 is showcased in Fig. 7. In terms of overall accuracy, we observe that the GNN model in Study 7 can maintains its performance by comparing with the performance of Study 6 (Fig. 6f). Also, by looking at the confusion matrix in Fig. 7, five out of six categories' sensitivities are around or above 95%, and the remaining Email category also has a sensitivity of 91.6%. According to the quantitative results, the GNN demonstrates its effectiveness and broad application for various classification tasks.

C. Hybrid Dataset Results:

1) Study 8 — Function-type Classification: The GNN model in Study 8 was trained using both VPN-dataset and NonVPN-dataset. Fig. 8 shows the confusion matrices derived from Study 8, which were created by testing the GNN only using the validation sets of VPN-dataset, NonVPN-dataset, and a joint dataset of both, respectively. The performance metrics of Study 8 is tabulated in Table V. By comparing Study 8 that the GNN model is validated only on VPN-dataset with Study 6 (Fig. 6f), the red boxes from Study 8 (Fig. 8a) shows

TABLE IV
PERFORMANCE METRICS OF STUDY 7

VPN-dataset	Overall Accuracy	Metric	Facebook	Hangouts	Skype	Email	Torrent	Voipbuster	Overall
Study 7	97.33%	Sensitivity	0.953	0.985	0.947	0.916	0.966	0.998	96.13%
		Precision	0.962	0.985	0.946	0.851	0.983	0.996	95.43%
		F1 score	0.958	0.985	0.947	0.882	0.974	0.997	95.76%



(a)	Chat	Email	File	Streaming	P2P	VoIP
Chat	97.6% 1569	6.7% 8	4.5% 17	1.7% 4	1.1% 2	0.7% 19
Email	0.9% 14	92.4% 110	0.8% 3	0.0% 0	0.0% 0	9.6% 270
File	0.8% 13	0.8% 1	87.6% 331	0.4% 1	0.6% 1	0.8% 22
Streaming	0.1% 2	0.0% 0	2.4% 9	95.7% 223	0.6% 1	0.1% 4
P2P	0.1% 1	0.0% 0	0.0% 0	0.0% 0	97.2% 174	0.0% 0
VoIP	0.6% 9	0.0% 0	4.8% 18	2.1% 5	0.6% 1	88.8% 2496

(b)	Chat	Email	File	Streaming	P2P	VoIP
Chat	87.4% 776	1.1% 27	1.9% 11	0.6% 4	0.0% 0	2.8% 51
Email	6.9% 61	98.2% 2373	3.1% 18	0.6% 4	0.0% 0	3.1% 55
File	1.4% 12	0.1% 2	70.8% 414	1.2% 8	3.5% 11	4.7% 85
Streaming	0.5% 4	0.1% 3	5.0% 29	91.7% 600	0.3% 1	2.2% 39
P2P	0.0% 0	0.0% 0	2.7% 16	0.0% 0	83.3% 265	2.1% 37
VoIP	3.9% 35	0.5% 11	16.6% 97	5.8% 38	12.9% 41	85.1% 1524

(c)	Chat	Email	File	Streaming	P2P	VoIP
Chat	94.0% 2345	1.4% 35	2.9% 28	0.9% 8	0.4% 2	1.5% 70
Email	3.0% 75	97.9% 2483	2.2% 21	0.5% 4	0.0% 0	7.1% 325
File	1.0% 25	0.1% 3	77.4% 745	1.0% 9	2.4% 12	2.3% 107
Streaming	0.2% 6	0.1% 3	3.9% 38	92.8% 823	0.4% 2	0.9% 43
P2P	0.0% 1	0.0% 0	1.7% 16	0.0% 0	88.3% 439	0.8% 37
VoIP	1.8% 44	0.4% 11	11.9% 115	4.8% 43	8.5% 42	87.4% 4020

Fig. 8. Hybrid dataset: Function-type classification. Confusion matrices for Study 8 using various validation datasets derived from: (a) VPN-dataset, (b) NonVPN-dataset, and (c) both VPN-dataset and NonVPN-dataset.

the sensitivities of Email, File, and VoIP have a decrease of 4.2%, 3.9%, and 9.9% respectively. However, the sensitivities of rest of the categories still remain high. The quantitative results show the GNN model in Study 8 can classify Function-type of network traffic flows in VPN-dataset effectively even when the GNN model is trained using both encrypted and non-encrypted traces. As for Fig. 8b where the GNN model is validated only on NonVPN-dataset, although the sensitivities of all the categories except for Email are relatively lower than the one validated on VPN-dataset (Fig. 8a), the features of non-encrypted traces still can be preserved in our trained GNN model to a certain extent.

2) Study 9 — Application-type Classification: The GNN model in Study 9 was trained using both VPN-dataset and NonVPN-dataset, and the performance metrics of Study 9 is tabulated in Table VI. Fig. 9a, Fig. 9b, and Fig. 9c are the confusion matrices derived from Study 9, which were created by testing the GNN using the validation sets of VPN-dataset, NonVPN-dataset, and a joint dataset of both, respectively. By comparing Study 9 that the GNN model is validated only on VPN-dataset with Study 7 (Fig. 7), the pink boxes from Study 9 (Fig. 9a) shows the sensitivities of Skype, Email, and Voipbuster have a decrease of 7.5%, 3.4%, and 6.5% respectively, while the sensitivities of Facebook, Hangouts, and Torrent still remain high. Also, similar to the trend occurring in Study 8, as shown in Fig. 9b, the GNN model in Study 9 has lower sensitivities of all the categories when it is validated only on NonVPN-dataset instead of VPN-dataset (Fig. 9a). Overall, according to the quantitative results, the GNN model in Study 9 shows its efficacy in classifying Application-type even though the model is trained with a mix of encrypted and non-encrypted traces.

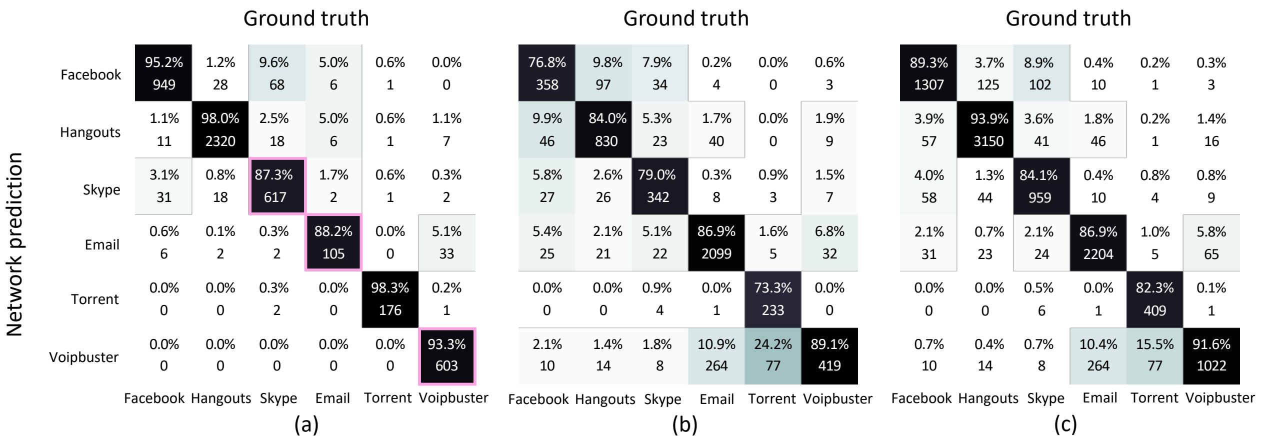
V. DISCUSSION

Study 6 proved that using GNNs for network traffic classification outperforms the two reference methods, the CNN-based and the LSTM-based model, using a VPN-dataset. We extracted VPN-dataset based on Function-type (i.e., Chat, Email, File, Streaming, P2P, and VoIP) rather than Application-type (i.e., Skype, YouTube, and Facebook). This increased the complexity of the classification problem, which aims to classify Function-type of network traffic flows across different applications. Additionally, the GNN model is trained with a hybrid of encrypted and non-encrypted dataset to demonstrate its efficacy in classifying both Function-type and Application-type. A key contribution of this work is to leverage a novel geometric learning framework [8] for predicting network traffic flows. While ordinary CNNs and LSTMs accept only raw bytes as network inputs, our GNN allows us to incorporate raw bytes, packet relations, and meta features that are assigned as node attributes, edge attributes and global attributes, respectively, to our network inputs. Importantly, our GNN accepts arbitrary graph inputs that contain any number of packets. Hence, regardless of memory, the GNN can obviate the needs of data truncation and zero padding, which are required by CNN-based and LSTM-based models. If memory consumption is constrained, it is possible to define a threshold for the amount of packets that will be fed into the GNN as a graph. In this scenario, the memory used by the GNN will be equivalent to or less than that required by the CNN or the LSTM models. Additionally, because the GNN can accept many relational-based inputs, investigations into potential factors in terms of packet relations need further exploration for robustness.

One potential limitation of our research is that the network traffic flows generated or captured by scripts are more likely

TABLE V
PERFORMANCE METRICS OF STUDY 8

Dataset	Overall Accuracy	Metric	Chat	Email	File	Streaming	P2P	VoIP	Overall
VPN-dataset	92.02%	Sensitivity	0.975	0.924	0.875	0.957	0.972	0.887	93.21%
		Precision	0.969	0.277	0.897	0.933	0.994	0.987	84.29%
		F1 score	0.972	0.426	0.886	0.944	0.983	0.934	85.80%
NonVPN-dataset	89.48%	Sensitivity	0.873	0.982	0.707	0.917	0.833	0.850	86.09%
		Precision	0.893	0.945	0.778	0.887	0.833	0.872	86.83%
		F1 score	0.883	0.963	0.741	0.902	0.833	0.861	86.42%
Overall	90.61%	Sensitivity	0.939	0.979	0.773	0.927	0.883	0.873	89.62%
		Precision	0.942	0.853	0.826	0.899	0.890	0.940	89.23%
		F1 score	0.941	0.912	0.799	0.913	0.886	0.905	89.31%



(a)

	Facebook	Hangouts	Skype	Email	Torrent	Voipbuster
Facebook	95.2% 949	1.2% 28	9.6% 68	5.0% 6	0.6% 1	0.0% 0
Hangouts	1.1% 11	98.0% 2320	2.5% 18	5.0% 6	0.6% 1	1.1% 7
Skype	3.1% 31	0.8% 18	87.3% 617	1.7% 2	0.6% 1	0.3% 2
Email	0.6% 6	0.1% 2	0.3% 2	88.2% 105	0.0% 0	5.1% 33
Torrent	0.0% 0	0.0% 0	0.3% 2	0.0% 0	98.3% 176	0.2% 1
Voipbuster	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	93.3% 603

(b)

	Facebook	Hangouts	Skype	Email	Torrent	Voipbuster
Facebook	76.8% 358	9.8% 97	7.9% 34	0.2% 4	0.0% 0	0.6% 3
Hangouts	9.9% 46	84.0% 830	5.3% 23	1.7% 40	0.0% 0	1.9% 9
Skype	5.8% 27	2.6% 26	79.0% 342	0.3% 8	0.9% 3	1.5% 7
Email	5.4% 25	2.1% 21	5.1% 22	86.9% 2099	1.6% 5	6.8% 32
Torrent	0.0% 0	0.0% 0	0.9% 4	0.0% 1	73.3% 233	0.0% 0
Voipbuster	2.1% 10	1.4% 14	1.8% 8	10.9% 264	24.2% 77	89.1% 419

(c)

	Facebook	Hangouts	Skype	Email	Torrent	Voipbuster
Facebook	89.3% 1307	3.7% 125	8.9% 102	0.4% 10	0.2% 1	0.3% 3
Hangouts	3.9% 57	93.9% 3150	3.6% 41	1.8% 46	0.2% 1	1.4% 16
Skype	4.0% 58	1.3% 44	84.1% 959	0.4% 10	0.8% 4	0.8% 9
Email	2.1% 31	0.7% 23	2.1% 24	86.9% 2204	1.0% 5	5.8% 65
Torrent	0.0% 0	0.0% 0	0.5% 6	0.0% 1	82.3% 409	0.1% 1
Voipbuster	0.7% 10	0.4% 14	0.7% 8	10.4% 264	15.5% 77	91.6% 1022

Fig. 9. Hybrid dataset: Application-type classification. Confusion matrices for Study 9 using various validation datasets derived from: (a) VPN-dataset, (b) NonVPN-dataset, and (c) both VPN-dataset and NonVPN-dataset.

TABLE VI
PERFORMANCE METRICS OF STUDY 9

Dataset	Overall Accuracy	Metric	Facebook	Hangouts	Skype	Email	Torrent	Voipbuster	Overall
VPN-dataset	95.10%	Sensitivity	0.951	0.979	0.872	0.882	0.983	0.933	93.39%
		Precision	0.902	0.981	0.919	0.709	0.983	1.000	91.60%
		F1 score	0.926	0.980	0.895	0.786	0.983	0.965	92.30%
NonVPN-dataset	84.09%	Sensitivity	0.768	0.840	0.789	0.868	0.732	0.891	81.52%
		Precision	0.721	0.875	0.828	0.952	0.979	0.529	81.43%
		F1 score	0.744	0.857	0.808	0.908	0.838	0.664	80.35%
Overall	89.55%	Sensitivity	0.893	0.938	0.841	0.869	0.822	0.915	88.02%
		Precision	0.844	0.951	0.884	0.937	0.980	0.732	88.85%
		F1 score	0.868	0.945	0.862	0.902	0.895	0.814	88.11%

to display deterministic behavior. Another limitation is that the trained GNN may not be applied to inputs with traffic flow characteristics (e.g., encryption protocols) that differ significantly from those used for training. This is a drawback shared by the majority of supervised learning models. Only VPN-dataset and NonVPN-dataset are used to train and validate the GNN. When the input flows are based on a different encryption protocol than the datasets used for training, it is anticipated that the GNN's performance may decrease or become biased. In our future work, we'll characterize how the GNN performs under various encryption protocols. One possible approach is to utilize transfer learning to retrain the GNN with a much smaller training dataset comprising flow inputs with different encryption methods. Another more sophisticated strategy is to use domain adaptation via adversarial training [26], which aims to train a neural network on a source dataset and achieve high accuracy on a target dataset that differs significantly from the source dataset. In this case, the task of the source and target domains is the same, but the source and target domains, data representation, or distribution are different.

VI. CONCLUSION

In this work, we proposed and implemented a flow-based geometric learning model for the classification of encrypted network traffic. We introduced the concept of mapping network traffic flows to graph representations, where data integrity can be preserved more effectively than when mapping from the original data. GNN models demonstrated superiority in executing multi-class classification on encrypted network traffic flows by utilizing packet relationships, combining raw bytes and meta features as input graphs, and overcoming the constraint of using the same data size. Then, in nine independent studies, we compared various combinations of network inputs and classification tasks for our GNN model. In terms of overall accuracy and F1 score metrics, the results show that our GNN, which utilizes raw bytes, meta features, and packet relations as network inputs, outperforms other GNN variants and the two reference methods. As part of our future work, for robustness, we will further validate our GNN model using a larger dataset that contains flows with various encryption protocols, investigate more useful and meaningful packet relation information, and study possible metrics for a comprehensive assessment of our GNN model in multi-class classification tasks.

Acknowledgments

The authors would like to thank the anonymous reviewers. This work is supported by NSF No.1738965 and a grant from Intel Corporation.

References

[1] R. T. El-Maghraby, N. M. Abd Elazim, and A. M. Bahaa-Eldin, "A Survey on Deep Packet Inspection," in *2017 12th International Conference on Computer Engineering and Systems (ICCES)*. IEEE, 2017, pp. 188–197.

[2] 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.

[3] 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.

[4] 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.

[5] O. Barut, R. Zhu, Y. Luo, and T. Zhang, "TLS Encrypted Application Classification Using Machine Learning with Flow Feature Engineering," in *2020 the 10th International Conference on Communication and Network Security*, 2020, pp. 32–41.

[6] M. Jaber, R. G. Cascella, and C. Barakat, "Can we trust the inter-packet time for traffic classification?" in *2011 IEEE international conference on communications (ICC)*. IEEE, 2011, pp. 1–5.

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

[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] 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.

[10] 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.

[11] J. Zhang, F. Ye, and Y. Qian, "Intelligent and Application-Aware Network Traffic Prediction in Smart Access Gateways," *IEEE Network*, vol. 34, no. 3, pp. 264–269, 2020.

[12] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$deep-full-range$: A Deep Learning Based Network Encrypted Traffic Classification and Intrusion Detection Framework," *IEEE Access*, vol. 7, pp. 45 182–45 190, 2019.

[13] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Computer networks*, vol. 165, p. 106944, 2019.

[14] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.

[15] 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.

[16] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate Decentralized Application Identification via Encrypted Traffic Analysis Using Graph Neural Networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2367–2380, 2021.

[17] B. Pang, Y. Fu, S. Ren, Y. Wang, Q. Liao, and Y. Jia, "CGNN: Traffic Classification with Graph Neural Network," *arXiv preprint arXiv:2110.09726*, 2021.

[18] T.-L. Huoh, Y. Luo, and T. Zhang, "Encrypted Network Traffic Classification Using a Geometric Learning Model," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 376–383.

[19] T. Shapira and Y. Shavitt, "FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1218–1232, 2021.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1180–1189. [Online]. Available: https://proceedings.mlr.press/v37/ganin15.html

VII. Biography Section

Ting-Li Huoh is currently pursuing the Ph.D. degree in Computer Engineering at University of Massachusetts Lowell. She received her first M.S. degree in Technology Management from National Chengchi University in 2018, and obtained a second M.S. degree in Computer Engineering from University of Massachusetts Lowell in 2022. Her research interests include computer networking, deep learning, big data analytics, blockchain technology, and artificial intelligence with a focus on network traffic analysis and network traffic classification.

Dr. Yan Luo is a Professor of the Department of Electrical and Computer Engineering at University of Massachusetts Lowell (UMass Lowell). He obtained his Ph.D. in Computer Science from University of California Riverside in 2005. His current research interest centers on machine/deep learning based data analytics of network flows and biomedical data, heterogeneous computing, and embedded sensor systems. He has served on the editorial board of several peer-reviewed journals and as a program chair of several IEEE conferences. Dr. Luo directs the laboratory of Advanced Computer Architecture and Network Systems (ACANETS) at UMass Lowell. He is a member of IEEE and ACM.

Dr. Peilong Li is an Assistant Professor of the Department of Computer Science at Elizabethtown College. He obtained his Ph.D. degree in Computer Engineering from University of Massachusetts Lowell in 2016. His research interests include heterogeneous and parallel computer architecture, big data analytics with distributed computing framework, and data plane innovation in software defined networking.

Dr. Tong Zhang is a Principal Engineer and Lead Architect on AI and analytics in the Network Platforms Group of Intel Corporation. She has been leading the architecture efforts for Machine Learning and Deep Learning solutions for network domain use cases. Before joining Intel, Dr. Zhang was a Principal Scientist in Hewlett-Packard Labs where she led over a dozen projects related to media analytics systems and products. She has over 50 granted US patents and more than 70 publications to her credit. She earned B.S. degree from Tsinghua University and Ph.D. degree from University of Southern California, both in Electrical Engineering.