GNN-Geo: A Graph Neural Network-based Fine-grained IP geolocation Framework

Shichang Ding, Xiangyang Luo*, Jinwei Wang, *Member, IEEE,* Xiaoming Fu, *Fellow, IEEE*

Abstract—Rule-based fine-grained IP geolocation methods are hard to generalize in computer networks which do not follow hypothetical rules. Recently, deep learning methods, like multi-layer perceptron (MLP), are tried to increase generalization capabilities. However, MLP is not so suitable for graph-structured data like networks. MLP treats IP addresses as isolated instances and ignores the connection information, which limits geolocation accuracy. In this work, we research how to increase the generalization capability with an emerging graph deep learning method – Graph Neural Network (GNN). First, IP geolocation is re-formulated as an attributed graph node regression problem. Then, we propose a GNN-based IP geolocation framework named GNN-Geo. GNN-Geo consists of a preprocessor, an encoder, messaging passing (MP) layers and a decoder. The preprocessor and encoder transform measurement data into the initial node embeddings. MP layers refine the initial node embeddings by modeling the connection information. The decoder maps the refined embeddings to nodes' locations and relieves the convergence problem by considering prior knowledge. The experiments in 8 real-world IPv4/IPv6 networks in North America, Europe and Asia show the proposed GNN-Geo clearly outperforms the state-of-art rule-based and learning-based baselines. This work verifies the great potential of GNN for fine-grained IP geolocation.

Index Terms—Fine-grained IP geolocation, Graph Neural Network, Computer Network, Deep Learning.

1 Introduction

IP geolocation is to obtain the geographic location (geolocation) of an IP address [1, 2]. It is widely used in localized online advertisement, location-based content restriction, cybercrime tracking, and so on [1, 3, 4]. IP geolocation is important especially for networking devices without GPS (Global Positioning System) function, such as PCs and routers. IP geolocation can be categorized into two kinds by accuracy: coarse-grained IP geolocation and fine-grained IP geolocation. Coarse-grained IP geolocation can find the city or state location of an IP address. Researchers have proposed a series of coarse-grained IP methods such as [1, 3]. The median error distances of these methods are usually between dozens and several hundreds of kilometers. After obtaining a coarse-grained location of a target IP, fine-grained IP geolocation methods can be used to find its more accurate location. The median error distances of fine-grained IP geolocation methods are usually less than 10 kilometers. Though there are several fine-grained IP geolocation methods like SLG [5], Checkin-geo [6], Corr-SLG [2] and MLP-Geo [7], they stills face many challenging problems. **In this paper, we mainly focus on improving the generalization capabilities of measurement-based fine-grained IP geolocation in different computer networks**.

Measurement-based IP geolocation can geolocate a target IP after the probing host [2] obtains the measurement data (e.g., delay) to the target IP and landmarks [2, 5]. Landmarks are IP addresses with known locations [2]. Generally, existing measurement-based fine-grained IP geolocation methods can be categorized into two kinds: rule-based and deep learning-based methods (referred as learning-based methods in this paper). For rule-based methods like SLG [5] and Corr-SLG [2], they assume that most hosts in one computer network are following some delay-distance rules. These hypothetical rules are proposed by human experts based on their observations. Then a target IP can be mapped to its geolocation based on the delay-distance rules. The performances of rule-based methods rely on the partition of hosts which follow the hypothetical rules. **This limits the generalization capabilities of rule-based methods in networks where hosts don't follow their hypothetical rules**.

Recently, deep learning seems to be a hopeful solution to the generalization problem of IP geolocation methods. There are two possible advantages of learning-based IP geolocation. First, learning-based methods do not rely on hypothetical rules which are suitable for all computer networks. They can learn a "measurement data - locations" relationship suitable to a specific computer network based on this network's raw measurement data. Second, learning-based methods are capable of extracting non-linear relationships, while non-linear rules are hard for human experts to observe. Encouraged by these possible advantages, researchers proposed several IP geolocation methods based on MLP (multi-layer perceptron), such as NN-Geo [8] and MLP-Geo [7]. Compared with NN-Geo, MLP-Geo is more accurate because it leverages not only delay but also routing information.

However, it is hard for MLP to model the connection information of computer networks. MLP is not good at modeling non-Euclidean structured data. Nevertheless, computer networks are fundamentally represented as graphs, a typical non-Euclidean structured data [9, 10]. MLP can only treat IP addresses as isolated instances, though hosts are non-independent and linked to each

- *Shichang Ding and Xiangyang Luo are with State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 276800, China.*(Corresponding author: Xiangyang Luo)
  E-mail: scdingwork@outlook.com, luoxy_ieu@sina.com
- *Jinwei Wang is with School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing 210044, China.*
  E-mail: wjwei_2004@163.com
- *Xiaoming Fu is with Institute of Computer Science, University of Göttingen, Göttingen 37077, Germany.*
  E-mail: fu@cs.uni-goettingen.de

Manuscript received , ; revised , .

other through intermediate routers. This is why MLP-Geo can only leverage the router IDs (and the delay) between the probing host and the targets, ignoring all the other useful features, such as the links and delays between routers. Based on the experiments (shown in Table 1), MLP-Geo could be less accurate than rule-based methods in some actual networks. **This indicates that IP geolocation needs a deep learning method more suitable for graph-structured data**.

Graph Neural Network (GNN) is an emerging deep learning method for graph-structured data presentation [11]. Recently, it has been successfully applied in different fields of graph-structured data, like recommendation system [12], social network [13], knowledge map [14], etc. However, little attention has been paid to introducing GNN into IP geolocation. As far as we know, this is the first work to explore the potential of GNN in fine-grained IP geolocation. The main challenges of introducing GNN into IP geolocation tasks are at least three-folds.

1) How to formulate IP geolocation problem in the context of graph deep learning?
2) How to design a GNN-based Framework for the fine-grained measurement-based IP geolocation task?
3) Which factors would clearly affect the performance of GNN in fine-grained IP geolocation performance?

By tackling these challenges, the main contributions of our work can be summarized as follows.

1) **Formulating IP geolocation as a graph learning task**. We formulate the measurement-based fine-grained IP geolocation task as a semi-supervised attributed-graph node regression problem. First, IP addresses are mapped into graph nodes, and links are mapped into graph edges. Then, the features of IP addresses (e.g., delay) and links (e.g., the delay of link) are transformed into attributes of the graph nodes and edges. Finally, the IP geolocation problem is formulated as estimating the latitudes and longitudes of the target nodes based on the known locations of other nodes as well as the topology and attributes of the graph. This paves the road of introducing graph deep learning methods into IP geolocation.
2) **GNN-Geo framework**. We propose a GNN-based framework (GNN-Geo) to solve the node regression problem. It consists of four components: a preprocessor, an encoder, MP (message passing) layers, and a decoder. The preprocessor transforms the raw traceroute data of a computer network into an attributed-graph representation. The encoder generates the initial graph node/edge embeddings for MP layers. MP layers refine the node embeddings by propagating messages along the edges. Finally, the latitudes and longitudes of each node are decoded from the refined node embeddings.
3) **Improving MLP-based decoder with rules**. We highlight the limitation of the vanilla MLP-based decoder, which makes GNN model hard to converge. The convergence problem is relieved with an idea like "rule-based" methods: making GNN-Geo to find targets' fine-grained locations inside a possible coarse-grained area instead of the whole earth. Experiments show that this significantly reduces the error distances and training epochs of GNN-Geo. Take Hong Kong as an example, compared with the vanilla MLP-based decoder, the improved decoder reduces the error distance from more than 2,690 km to less than 8 km.
4) **Large-scale experiment results outperform the best baselines**. The experiments are carried out in 8 real-world IPv4/IPv6 networks including New York State, Hong Kong, Shanghai, Beijing, Tokyo, Tokyo (IPv6), Berlin (IPv6) and Munich (IPv6). The results show: (i) when the training dataset ratio is 70%, GNN-Geo outperforms the best baselines averagely by more than 21% w.r.t. both average error distance and median error distance; (ii) When training dataset ratio decrease from 70% to 10%, though its advantages are reduced on sparser dataset, GNN still outperform baselines in most cases. We also analyze how different factors (e.g., basic GNN models, GNN aggregation methods) would affect GNN-Geo's performance. These results validate the value of introducing GNN in fine-grained IP geolocation.

The rest of the paper is organized as follows. Related works are introduced in Section 2. In Section 3, IP geolocation is formulated into a graph node regression problem. Then in Section 4, we discuss why GNN is worth to be tried in IP geolocation. In Section 5, we introduce the detail of proposed GNN-Geo. Section 6 shows and analyzes the experiment results in three real-world datasets. Section 7 discusses several possible future improvements for GNN-Geo. Finally, the paper is concluded in Section 8.

2 RELATED WORK

In this section, we introduce the works related to our study, mainly including fine-grained IP geolocation and Graph Neural Network.

2.1 Fine-grained IP geolocation

Besides measurement-based methods, there are two other kinds of fine-grained IP geolocation methods: database and data-mining-based method. IP geolocation databases can only provide the fine-grained locations for a small proportion of IP addresses, which could hardly satisfy people's needs [15]. Data-mining-based methods, like Checkin-Geo [6], can map IP addresses to locations based on a large amount of raw data which contains "IP-location" information such as users' logs. However, these kinds of raw data resources are only possessed by few internet giants like Microsoft Bing [4, 15] and Tencent [6], which are not open to the public. For example, Checkin-Geo is a data-mining-based fine-grained method. First, it extracts users' names and their locations from the checkin [6] information collected on smartphone applications. Then it extracts users' names and users' IP addresses from the logins collected on PC applications. In the end, Checkin-Geo can map a user's PC IP address to his/her smartphone location. Both the PC and the smartphone applications are owned by a famous Chinese online social platform called "Tencent QQ". Recently, researchers from Microsoft build a large-scale IP-location ground-truth dataset, which is mined from the query logs of a commercial search engine. Based on the large-scale dataset, they can use machine learning methods to estimate the location of IP addresses from their reverse DNS hostnames [4]. We can see, these high quality large-scale ground truth data resources are hard to get for researchers outside Tencent or Microsoft. Besides this problem, these methods cannot work well in areas where those Internet giants are not serving, or the location-sharing service is forbidden. For example, Tencent QQ mainly serves in China and has been

banned in India[1].

The limitations of database and data-mining-based methods require researchers to pay attention to measurement-based methods. In theory, measurement-based methods can geolocate any target IP if the delay or routing data between the target IP address and landmarks can be obtained. However, it is hard for measurement-based methods to get a fine-grained geolocation result. Because the relationship between measurement data and target IP's location is very complicated and different in various computer network environments.

SLG is the first fine-grained measurement-based IP geolocation method [5]. It is also the first rule-based method as well as one of the most widely-used fine-grained IP geolocation baselines until now. It assumes that most hosts in a computer network are following a simple linear delay-distance rule: the shortest "relative delay" comes from the nearest landmark. Since the delay between a landmark and a target IP is hard to measure, "relative delay" is proposed by SLG as an approximation. Assume the delay from the probing host to a target IP is d_{pt}, the delay from the probing host to a landmark is d_{pl}, and the delay from the probing host to the closest common router [2, 5] shared by the target IP and landmark is d_{pr}, the "relative delay" between the target and the landmark is $(d_{pt}-d_{pr})+(d_{pl}-d_{pr})$. This rule originated from an observation of the relationship between relative delay and distance of 13 landmarks in New York City [5]. However, in further studies, researchers realized that this rule may be not valid for all intra-city networks [2].

Some researchers find that there is no clear relationship between relative delay and distance in cities like Zhengzhou (China) and Toronto (Canada) [2]. They explain that for the intra-city delay, the delay caused by geographical distance are not always the major constituent. The other factors like queuing delay and processing delay in routers could be more important. Thus the relationship between relative delay and distance is very complicated. In Corr-SLG, the relative-delay-distance correlation of all landmarks is calculated at first. Then all landmarks are divided into three groups. Corr-SLG leverages different delay-distance rules in each group. In the groups with a strong-positive relative-delay-distance correlation (close to 1), the relative delay increases as the distance increases. Thus the rule is similar to SLG, and Corr-SLG still map targets to the landmarks with the shortest relative delay. In the groups with a strong-negative correlation, they map targets to the landmarks with the largest relative delay (close to -1). However, Corr-SLG cannot decide how to handle the groups with weak correlation (close to 0). They simply map these targets to the average locations of landmarks. We can see the accuracy of the rule-based methods relies on the partition of hosts which follows their pre-assumed rules. Since the network environments may change in each city and at different times, the relationships between delay and distance may vary significantly. Thus, the performances of rule-based methods vary significantly. **How to design a fine-grained method that can well generalize in different networks is an important problem**.

In recent years, deep learning has shown clear advantages in various networking problems [16–21]. People also begin to try deep learning methods in IP geolocation to improve the generalization capabilities. Both NN-Geo [8] and MLP-Geo [7] use MLP to estimate locations for target IP addresses. Compared with NN-Geo, MLP-Geo adds a new kind of useful information – the routers' IDs between probing hosts and the targets. Its performance is clearly better than NN-Geo since NN-Geo only utilizes the delay between probing hosts and the target IP. The aim of learning-based methods is not to find a rule which is suitable for all networks. Instead, they aim to present a method which can correctly find "rules" in different networks. Whether a method can find rules correctly mainly relies on its modeling ability towards the target networks. Actually, MLP is not so suitable for modeling the measurement data of computer networks. MLP is good at prepossessing Euclidean structure data. However, the topology of the internet is a non-Euclidean structure data [22]. MLP can only treat target IP addresses as isolated data instances while ignoring the connection information between targets. This would easily lead to suboptimal representations and performance impairment. We still need to improve the learning-based methods by utilizing deep learning methods more suitable to networks.

2.2 Graph Neural Network (GNN)

Graph Neural Network (GNN) is an emerging deep learning method, which is specially designed for graph-structured data [11, 23, 24]. GNN learns the representation of a graph node by recursively aggregating the information of its neighbor nodes. Then, the learned node representations can be used for node-level, edge-level or graph-level classification/regression tasks. There are several typical GNN architectures such as GCN (Convolutional Neural Network) [11], GAT (Graph Attention Network) [23], MPNN (Message Passing Neural Network) [24], etc. Most existing GNN-based methods can be seen as their variants or improved version. GCN is inspired by CNN's success in image processing area [25]. GCN uses multiple graph convolutional layers to aggregate information from neighbor nodes. GAT introduces attention mechanisms into information aggregation process. GCN usually aggregates the normalized sum of neighbors' information while GAT assigns attention weights to neighbors to get a weighted sum of neighbors' information. This can help GAT to differentiate the contributions of different neighbors and gain better performance. Both GCN and GAT mainly focus on aggregating node feature and ignore edge feature [26]. However, edge feature is crucial for IP geolocation tasks. For example, the delay or hop count between IP addresses have been verified as very important features in IP geolocation in previous methods [2, 5]. Compared with GCN and GAT, MPNN is good at processing different kinds of edge features (including categorical and continuous). Thus in this paper, we use MPNN as the basic GNN model for the GNN-based IP geolocation framework.

Due to its powerful spatial modelling ability, GNN has achieved impressive performances in various graph-type data areas, such as social network [27], knowledge map [28] and recommendation system [29]. Especially, there is also a kind of geolocation task in social network – user geolocation. User geolocation (UG) is to identify the geographic location of online social network (OSN) users, such as Twitter users [27]. Several works such as [27, 30–32] have shown GNN's advantages in user geolocation. For example, [27] presents Hierarchical Graph Neural Networks (HGNN) which leverages robust signals from geographically close crowds rather than individuals. Through exploiting both unlabeled nodes and isolated nodes, HGNN achieves better performance than traditional learning-based methods. Though computer network and OSN are both represented as

1. www.xda-developers.com/india-bans-xiaomi-mi-browser-pro-qq-international-app/

graph-structured data, GNN-based UG methods cannot be simply transferred to IP geolocation. First, OSNs like Twitter usually contain many kinds of node features like user posts, location tags, pictures while node features in IP geolocation are quite limited, mainly relies on network measurement data (such as delay). Second, the links between OSN users usually do not contain crucial features like delay or hop count in IP geolocation. Thus, existing GNN-based UG methods usually ignore edge features and focus on exploiting rich node features. These GNN methods may lose important information when transferred to IP geolocation tasks. GNN-based IP geolocation methods should pay attention to both node and edge features. Besides, the accuracy of existing UG methods is mainly around city-level while existing fine-grained IP geolocation methods have already achieved street-level. The newly proposed GNN-based fine-grained IP geolocation methods should also achieve street-level.

Recently, researchers also begin to try GNN in different problems of computer networks [22]. For example, network modeling is used to reconstruct the computer networks and predict the structure/topology of unseen computer networks. GNN has been introduced into network modelling in [33, 34]. [35, 36] discuss how to utilize GNN in network calculus analysis. GNN is also helpful in link delay prediction [37] and network traffic prediction [38–40]. GNN has been introduced into automatic detection for Botnets, which is important to prevent DDoS attacks[41]. However, most previous GNN-based works in computer networks are focusing graph-level or edge-level classification or regression tasks. Little attention has been paid to how to model the relationship between location signals from network measurement data for IP geolocation (node regression/classification) until now.

3 IP GEOLOCATION FORMULATION

To explore the potential of GNN in fine-grained IP geolocation, we need to formulate IP geolocation as a graph-based learning problem at first. Before introducing the formulation, we want to clarify several issues here. Since there have been many mature coarse-grained IP geolocation methods, in this paper, we assume the coarse-grained locations of targets have been obtained before fine-grained geolocation. Thus, when mentioning a computer network in this work, it usually means a city-level network or at maximum state-level network. We also suggest researchers to select a most suitable coarse-grained method in each area. Since coarse-grained methods' performances may often change in different areas, and more accurate coarse-grained location results usually help a lot for following fine-grained IP geolocation activities.

In this paper, the IP addresses $IP_i, i = 1, 2, ..., N_{IP}$ and the links $LK_j, j = 1, 2, ..., N_{LK}$ between IP addresses are used to represent the topology of a computer network. The link means a direct physical link between two IP addresses. The location of each IP address is represented as a pair of (latitude, longitude). Each IP address is associated with measured attributes, such as the delay from the probing host. Each link is also associated with measured attributes, such as the delay of the link. All IP addresses can be divided into four groups: the probing host IP address, the landmark IP addresses, the target IP addresses and the router IP addresses. The router IP addresses are the intermediate router IP addresses found by the probing host when tracerouting the landmarks and targets.

As shown in Fig. 1, we map a computer network topology to an attributed undirected graph, $G = (V, E, XV, XE)$. In the graph, a graph node $v_i \in V, i \in \{0, 1, \ldots, N_V\}$ represents an IP address IP_i. Each link LK_j is represented as a graph edge $e_j \in E, j \in \{0, 1, \ldots, N_E\}$. Let $xv_i \in XV, i \in \{0, 1, \ldots, N_V\}$ and $xe_j \in XE, j \in \{0, 1, \ldots, N_E\}$ represent the attributes of the node v_i and the edge e_j, respectively. The latitude and longitude location of v_i are loc_{v_i}. The two dimensions of loc_{v_i} are $[-90, 90]$ and $[-180, 180]$, respectively. Assume the nodes belong to the probing host, landmarks, targets and routers are V_p, V_l, V_t and V_r. Then the locations of the probing host and landmarks are $\{loc_{v_i}, v_i \in \{V_p, V_l\}\}$. The locations of targets are $\{loc_{v_i}, v_i \in V_t\}$. Herein, the measurement-based IP geolocation problem is formulated to learn a prediction function $f(\cdot)$, which can estimate the targets' location:

$$[\{loc_{v_i}, v_i \in \{V_p, V_l\}\}; G] \overset{f(\cdot)}{\rightarrow} \{\widehat{loc_{v_i}}, v_i \in V_t\}, \quad (1)$$

where the outputs are targets' estimated locations $\{\widehat{loc_{v_i}}, v_i \in V_t\}$, and the inputs are (i) the attributed undirected graph $G = (V, E, XV, XE)$; (ii) the known locations of the probing host and landmarks $\{loc_{v_i}, v_i \in \{V_p, V_l\}\}$. In this paper, we use the numerical values of (latitude, longitude) to represent the location of each node, thus this is an attributed graph node regression problem. If the location is categorical, like communities or streets, the IP geolocation problem can also be formulated into a node classification problem. Note that researchers usually do not have the location information of intermediate routers V_r, thus this is also a typical semi-supervised problem. It is worth reporting here, it is also easy to estimate the location of intermediate routers by formulating IP geolocation into a graph node regression problem. Because the graph representations of router nodes can be naturally learned along with target nodes.

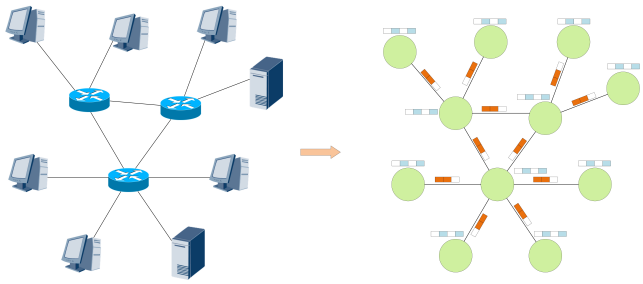


Fig. 1: Mapping a Computer Network into An Attributed Graph

4 THE POSSIBLE VALUE OF GNN FOR IP GEOLOCATION

Generally speaking, the two most important phases in learning-based IP geolocation methods are: (i) target embedding and (ii) target location mapping. From previous methods like [2, 5, 7], we know raw network measurement data contains useful signals which can help in IP geolocation. The target embedding phase is responsible to extract all useful signals from raw network measurement data, and then compress them into vectorized representations of target IP addresses (i.e., targets' embeddings). Then the target location mapping phase is responsible to map targets' embeddings to targets' locations. **The main problem of MLP-Geo is that its target embedding phase loses many useful signals from measurement data**.

MLP-Geo consists of three dense layers [7]. Here we assume its target embedding phase includes the first two layers. And its

target location mapping phase is the last layer. Then the targets' embeddings E_2^{mlp} of MLP-Geo are generated as follows:

$$E_2^{mlp} = f_2(f_1(Input)), \tag{2}$$

$$f_i = \sigma(W_i X_i + b_i), \tag{3}$$

where $Input$ is the input features of MLP-Geo, f_i denotes the i-th-dense-layer of MLP-Geo, W_i is the weight of the i-th-dense-layer, and b_i is the bias of the i-th-dense-layer. X_i is the input of i-th-dense-layer and the output of the $(i-1)$-th-dense-layer. $Input$ is X_0. In MLP-Geo, $Input$ consists of the delay and router IDs from the probing host to the targets.

From Formula 3, we can see the dense layer f_i uses matrix multiplication to extract useful signals from the input data. In this way, it can only judge whether a router ID shows in the path between the probing host and a target. MLP-Geo cannot extract the sequence relationship of the routers between the probing host and targets. It also cannot describe the topology around an IP addresses (or the whole network). And it cannot leverage the delay between two directly-linked IP addresses, which have been proved useful by the rule-based methods like SLG. It is hard for MLP-Geo to extract these useful signals because it is not designed for non-Euclidean structure data like graph.

The core concept of GNN (or more precisely, MPNN in this paper) is message passing [24]. As introduced in last section, all IP addresses are transformed into graph nodes. The target embedding phase of GNN is as follows:

$$E_i^{gnn} = f_i(\ldots(f_2(f_1(E_0)), \tag{4}$$

$$f_i = \{f_{message}, f_{aggregate}, f_{update}\}, \tag{5}$$

$$M_i = f_{message}(E_{i-1}^{neighbor}, E^{link}), \tag{6}$$

$$MS_i = f_{aggregate}(M_i), \tag{7}$$

$$E_i^{gnn} = f_{update}(MS_i, E_{i-1}^{gnn}), \tag{8}$$

where f_i indicates the i-th-MP-layer, E_i^{gnn} is the node embeddings of the i-th-MP-layer, E_0 denotes the input (initial node embeddings). We can see both GNN and MLP learn new embeddings for IP addresses over multiple layers. GNN is a general framework. One MP layer f_i consists of three functions: the message function $f_{message}$, the aggregate function $f_{aggregate}$ and the update function f_{update}. For a node, the message function $f_{message}$ passes its neighbor node's embedding ($E_{i-1}^{neighbor}$) along the edge/link (E^{link}) to it. A neighbor node means there is one edge between these two nodes. Then the aggregate function $f_{aggregate}$ collects the aggregation of all its neighbor nodes' messages (MS_i). The update function f_{update} forms the node's new embedding E_i^{gnn} based on the aggregated messages MS_i and the node's previous embeddings E_{i-1}^{gnn}.

From Formula 4-8, we can see the embedding of each graph node also aggregates with neighbor nodes' embeddings and its edge embeddings at the beginning of each MP layer. And in i-th-MP-layer, a node can get information from its i-hop neighbors. For example, in the $2nd$-MP-layer, a node not only gets messages from its 1-hop neighbors but also gets information from 2-hop neighbors. Because the messages from its 2-hop neighbors are passed to its 1-hop neighbors in $1st$-MP-layer. Through passing messages along the edges, after several MP layers, every node will gradually know the topology and node/edge attributes of the computer network. So compared with MLP, GNN can naturally describe the network topology in the targets' embeddings. And the other features like delays of links and IP addresses can also be utilized, because they can be easily combined into message function $f_{message}$, or the embeddings of links E^{link}. Thus, in some networks, if their measurement data (such as network topology, delays between routers and IP addresses) contain useful information for IP geolocation, and also is enough for GNN to learn (deep learning methods would overfit if dataset is too sparse), then GNN may show better performance in these networks. Next, to test how better GNN could be in real IP geolocation tasks, we design a GNN-based IP geolocation framework and carry out experiments to measure its performance in various real networks.

5 Our Proposed GNN-Geo Framework

We present the GNN-Geo framework in this section. Fig. 2 illustrates the framework, which consists of four components: a preprocessor, an encoder, MP layers, and a decoder. The preprocessor maps the raw measurement data of a computer network into its graph representation $G = (V, E, XV, XE)$. The encoder generates the initial feature embeddings of G. The MP layers refine the initial feature embeddings with graph signals and output the refined embeddings for all nodes V. Finally, the decoder maps the refined node embeddings into the locations. We will describe each component one by one, followed by the optimization details. It is worth mentioning that by changing some details of each component, we can get different GNN-based IP geolocation methods from the GNN-Geo framework.

5.1 The Preprocessor

In this paper, the measurement data is mainly traceroute data from the probing host to landmarks and targets. The task of the preprocessor is transforming the raw traceroute data into the initial embeddings of $G = (V, E, XV, XE)$ for the encoder. This mainly consists of two sub-tasks: (i) building the Graph G with the graph nodes V and the links E; (ii) extracting the node attributes XV and the link attributes XE.

5.1.1 Building Graph for the Measured Computer Network

Node. As mentioned in Section 3, we use the IP addresses and the links between IP addresses to represent the topology of a computer network. Thus all IP addresses in the raw traceroute data are transformed into the graph nodes. Each node v_i is differentiated by a node ID. The ID is from 1 to N_V. N_V is the number of all IP addresses found in traceroute data.

Edge. If the probing host finds a direct physical link between two IP addresses, then the link is transformed into a graph edge e_j. During the traceroute, it is possible that the probing host cannot get the IP addresses and delays of some routers, which are called "anonymous routers". There are two kinds of methods to build an edge over an "anonymous router": (i) ignore the anonymous router and build an edge between the two routers before and after the anonymous router along the routing path; (ii) map the anonymous routers to an IP address found in other re-measured routing paths. Both solutions are tested in our early experiments. The difference in their final geolocation performances is not significant. This may be due to that GNN itself is a powerful feature extractor, which is not sensitive to such differences in the input graph. However, the first solution introduces more edges in G, which leads to higher computation complexity and greater memory needs in the optimization phase. Thus we leverage the second solution in this paper, which is introduced as follows.

After tracerouting one IP address multiple times, we may get

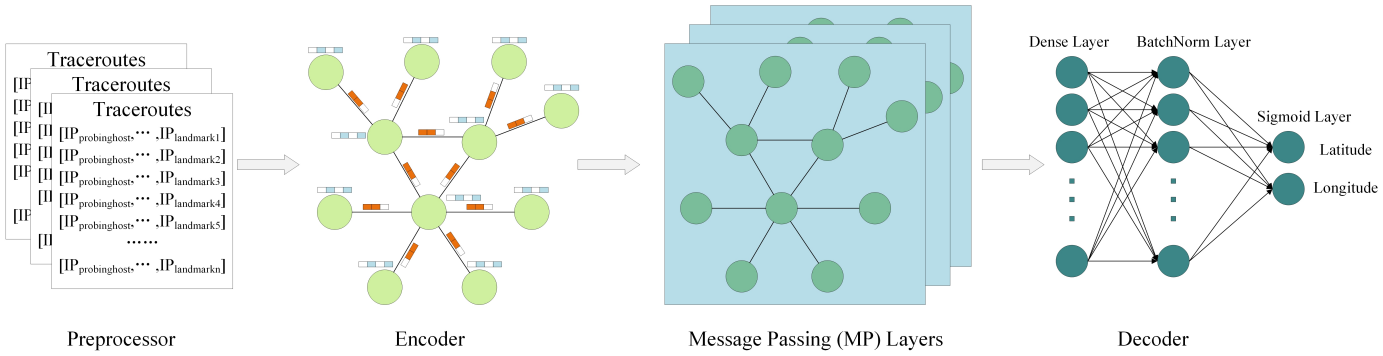


Fig. 2: Illustration of the GNN-Geo Framework

several different routing paths. We assume that if two routing paths are the same as each other except for the anonymous routers, then the two paths are likely to be similar. Then we can map the anonymous routers in one path to the IP addresses in other similar paths. Our matching algorithm is shown in Algorithm 1. The raw path set refers to all the raw paths found by the traceroute. The completed path set refers to the paths of which the anonymous routers have been mapped to IP as much as possible. After using Algorithm 1, we can build edges based on the completed path set. We ignore any anonymous routers which are still not mapped to IP addresses. And then an edge is built between the two routers before and after the anonymous routers along the routing path. The number of all final edges is N_E. It is worth mentioning here researchers can replace Algorithm 1 with other methods to check its influence on the final geolocation performance in future works.

Algorithm 1: Map Anonymous Routers to IP addresses

Input: Raw path set P_r, Completed path set $P_c = \varnothing$, Flag $flg = 1$
Output: Completed path set P_c

```
: for each P_r^i ∈ P_r do
    flg = 1;
    for each P_c^j ∈ P_c do
        if the hop count of P_r^i equals P_c^j then
            if the hop of P_r^i equals P_c^j except anonymous routers then
                Update anonymous routers in P_c^j with corresponding IP in P_r^i;
                Update P_c^j in P_c with updated P_c^j;
                flg = 0;
                break;
            end
        end
    end
    if flg = 1 then
        Add P_r^i into P_c;
    end
    else
        flg = 1;
    end
end
final;
return P_c
```

5.1.2 *Extracting Attributes*

Node Attributes. In this work, each graph node is associated with two kinds of node attributes: node delay and node IP address. The node delay is the direct delay from the probing host to the node. For each node, the delay is repeatedly measured by the probing host many times. The minimum one is selected as the node delay because it contains minimum congestion and is closer to the true propagation delay [2]. In the end, we combine the node IDs and node attributes as the initial feature of a node. $\mathbf{v_i}$ denotes the initial features of node v_i and $\mathbf{V}$ denotes the initial features of all nodes.

Edge Attributes. Each edge can be associated with several kinds of features such as the edge delay, the head node IP address, and the tail node IP address. Here, the node nearer the probing host is called the head node of the edge, and the other one is the tail node. The edge delay is calculated by subtracting the node delay of the head node from the tail node. We keep edge delay even if it could be a negative value. Since we do not need to refine the edge representation in the MP layers, we do not need edge IDs to differentiate edges. The initial features of an edge only consist of its edge attributes. $\mathbf{e_j}$ denotes the initial features of edge e_j and $\mathbf{E}$ denotes the initial features of all edges.

5.2 The Encoder

The encoder aims to form initial low dimensional embeddings of graph nodes and edges for the MP layers. The embeddings of graph nodes and edges are generated from the initial node features $\mathbf{V}$ and the initial edge features $\mathbf{E}$ from the preprocessor.

For each non-zero feature in $\mathbf{V}$ and $\mathbf{E}$, we associate it with an embedding vector. We concatenate the embeddings of a node (or edge) into a vector to describe the node (or edge). To be specific, the low dimensional embeddings of a node v_i and an edge e_u are:

$$\mathbf{h}_{v_i}^{(0)} = {\mathbf{Q_v}}^T \mathbf{v_i}, \tag{9}$$

$$\mathbf{h}_{e_j} = {\mathbf{Q_e}}^T \mathbf{e_u}, \tag{10}$$

where $\mathbf{Q_v} \in \mathbb{R}^{N_V \times G}$ denotes the embedding matrix for all nodes V, $\mathbf{Q_e} \in \mathbb{R}^{N_E \times K}$ denotes the embedding matrix for all edges E. N_V denotes the number of node features and G denotes the embedding size. N_E denotes the number of edge features and K denotes the embedding size.

5.3 Message Passing (MP) Layers

The inputs of Message Passing (MP) layers are $\mathbf{h}_V^{(0)}$ and $\mathbf{h}_E$ from the encoder. MP layers will augment the initial graph node embeddings ($\mathbf{h}_V^{(0)}$) by explicitly modeling the edges between nodes

and graph/edge attributes. Each MP layer consists of message, aggregate and update functions as follows.

Message Function. For a node v_i and one of its neighbour nodes v_j, the message from v_j to v_i is defined as:

$$\mathbf{m}_{i\leftarrow j} = f_{message}(\mathbf{h}_{v_j}, \mathbf{e}_{i\leftarrow j}), \tag{11}$$

The inputs of $f_{message}$ are: (i) the embedding of the neighbour node $\mathbf{h}_{v_j}$; (ii) the embedding of the edge $\mathbf{e}_{i\leftarrow j}$ from v_j to v_i. In this paper, $f_{message}$ is implemented as:

$$\mathbf{m}_{i\leftarrow j} = f_{edge}(\mathbf{h}_{e_{i\leftarrow j}})\mathbf{h}_{v_j}, \tag{12}$$

where the edge network f_{edge} is a two-layer MLP, which transforms the edge embedding $\mathbf{h}_{e_{i\leftarrow j}}$ to a matrix $W_{e_{i\leftarrow j}}$ for a neighbor node's message h_{v_j}. The architecture of f_{edge} is:

$$\mathbf{W}_{e_{i\leftarrow j}} = \mathbf{W}^2_{edge}(\mathrm{Relu}(\mathbf{W}^1_{edge}(e_{i\leftarrow j}))), \tag{13}$$

where $\mathbf{W}_{e_{i\leftarrow j}}$ is used as a weight to control the influence of the neighbor's message $\mathbf{h}_{v_j}$ to node i. $\mathbf{W}^1_{edge}$ and $\mathbf{W}^2_{edge}$ are the weight matrices of the two MLP layers. They first transform the initial edge embedding $\mathbf{h}_{e_{i\leftarrow j}}$ (size K) to a temporary embedding (size $2K$), then transform the temporary embedding into a one-dimensional embedding. The one-dimensional weight embedding will be reshaped as a two-dimensional square weight matrix $\mathbf{W}_{e_{i\leftarrow j}}$ (size $G \times G$) to control the influence of neighbor node v_j to node v_i. The activation function of ReLU [42] is leveraged to catch the non-linear relationships.

The edge embedding $\mathbf{h}_{e_{i\leftarrow j}}$ is generated from the edge attributes like the edge delay and the IP addresses of two edge nodes. $\mathbf{h}_{v_j}$ contains the node ID and attributes (e.g., IP addresses, the delay to the probing host). It is reasonable to use the transformed edge embeddings as message weight. For example, if the edge delay is too large, it may reflect that v_j is too far away, then its message is not so important to v_i; Or if the two IP addresses are very similar, it may reflect that the two nodes belong to one organization, then v_i should pay more attention to v_j. In this way, node v_i will learn what is v_j's information ($\mathbf{h}_{v_j}$), and can decide how much information it want to receive from the relationship ($f_{edge}(\mathbf{h}_{e_{i\leftarrow j}})$).

Aggregation and Update Function. U_{v_i} denotes the set of neighbor nodes of v_i. The aggregation function $f_{aggregate}$ collects all the messages propagated from U_{v_i} to v_i. The update function f_{update} uses the collected messages A_{v_i} to update the embedding of v_i. The aggregation and update function for v_i are defined as:

$$A_{v_i} = f_{aggregate_{v_j \in U_{v_i}}}(\mathbf{m}_{i\leftarrow j}), \tag{14}$$

$$\mathbf{h}^{(l)}_{v_i} = f_{update}(\mathbf{h}^{(l-1)}_{v_i}, A_{v_i}), \tag{15}$$

where $\mathbf{h}^{(l)}_{v_i}$ denotes the updated embedding of node v_i after the l-th MP layer ($l = 1, 2, 3, 4, \dots$). $\mathbf{h}^{(0)}_{v_i}$ is the initial node embedding from the encoder, as shown in Formula 9. The aggregation function $f_{aggregate}$ can be a simply symmetric function such as Mean [11], Max [43], or Sum [44]. Since no previous work has tested them in IP geolocation, we tried all three kinds of aggregators in experiments. The update function is implemented as follows:

$$\mathbf{h}^{(l)}_{v_i} = \mathrm{ReLU}(\mathbf{h}^{(l-1)}_{v_i} + A_{v_i}), \tag{16}$$

where we update the node v_i by summing up its previous embedding $\mathbf{h}^{(l-1)}_{v_i}$ and its aggregated messages from neighbors A_{v_i}. Relu is used to do non-linear feature transformation since nodes and edges have rich semantic attributes in this paper. In this way, the new node embedding $\mathbf{h}^{(l)}_{v_i}$ contains following information : (i) which nodes are its neighbors (topology nearby); (ii) its neighbors' attributes; (iii) its edges' attributes (e.g., IP address, delay); (iv) its own attributes (e.g., ID, IP address, delay). By stacking L MP layers, a node is capable of receiving the messages propagated from its L-hop neighbors. The final embeddings for all node $\mathbf{h}^{(L)}_V$ are then fed into the decoder for location estimation.

5.4 The Decoder

The decoder aims to estimate the locations from the nodes' refined embeddings $\mathbf{h}^{(L)}_V$. We need to predict two numerical values (latitude and longitude) of all nodes. For problems in other research areas, a widely-used original decoder after MP layers consists of dense layers. For example, a typical two-dense-layers decoder is:

$$\widehat{loc} = \sigma\left(\mathbf{W}^2_{loc}(\mathbf{W}^1_{loc}\mathbf{h}^{(L)}_V + b^1_{loc}) + b^2_{loc}\right), \tag{17}$$

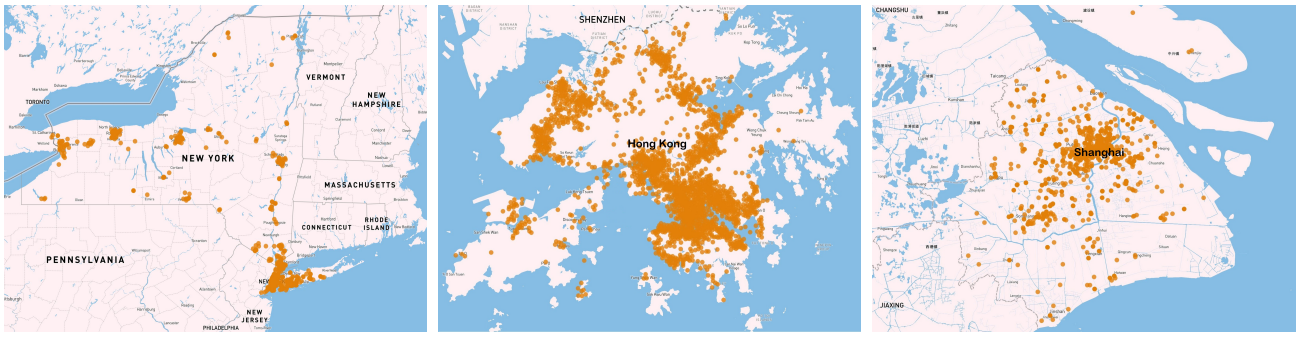
where $\widehat{loc} \in \mathbb{R}^{N_V \times 2}$ denotes the estimated location matrix for all nodes V. The two dimensions of $\hat{loc}$ are the estimated numerical values of latitude and longitude for all nodes. $\mathbf{W}^1_{loc}$ and $\mathbf{W}^2_{loc}$ are the weights of the two dense layers. b^1_{loc} and b^2_{loc} are biases. σ is the non-linear activation function to endorse nonlinearity, such as Relu in Formula 16. Then we can train GNN-Geo by comparing the ground-truth locations loc_{train} and the estimated locations $\hat{loc}_{train}$. loc_{train} (or $\hat{loc}_{train}$) means the real/estimated locations of the training dataset.

However, this kind of decoder is not so suitable for IP geolocation: the output can be any data in $\mathbb{R}$ while the latitude and longitude range are (90S~90N, 180W~180E). Thus most output data is useless and significantly increases the difficulty of optimization. Actually, even finding a fine-grained location of targets among the whole earth is not necessary. As mentioned in Section 1, fine-grained IP geolocation is on the basis of coarse-grained geolocation. Thus, we usually already know a rough location of the target before estimating its fine-grained location. Inspired by this, we combine rules into the pure MLP-based decoder to improve the performance of GNN-based IP geolocation methods. First, we need to use two min-max scalers to transform the loc_{train} (latitude & longitude) into loc^t_{train}. The scales of transformed latitude and longitude in loc^t_{train} are both $[0, 1]$. Then our decoder is implemented as follows:

$$\widehat{loc^{ts}} = \mathrm{Sigmoid}(\mathrm{BatchNorm}(\mathbf{W}_{loc}\mathbf{h}^{(L)}_V + \mathbf{b}_{loc})), \tag{18}$$

where $\widehat{loc^{ts}}$ is the estimated transformed location matrix for all nodes V. $\mathrm{Sigmoid}$ is an activation function of which output is $[0, 1]$. $\mathrm{Sigmoid}$ could become saturated if input values are too large, which makes learning even harder. $\mathrm{BatchNorm}$ refers to Batch Normalization [45]. Batch Normalization is applied here to ease the saturating problem of Sigmoid as well as preventing overfitting. Then we train GNN-Geo by comparing loc^{ts}_{train} and $\widehat{loc^{ts}_{train}}$. After training, $\widehat{loc}$ can be scaled from $\widehat{loc^{ts}}$ by reversely using two min-max scalers.

It is worth mentioning that we do not always need to scale the range of latitude and longitude of the training sets. The core idea is to find a rough area of targets, then re-scale the range of latitude and longitude of these areas to help the optimizer to converge. In this way, the decoder is actually based on a hypothetical rule



(a) New York State (b) Hong Kong (c) Shanghai

Fig. 3: Landmark Distribution of Three Areas as an example

that we could get a coarse-grained location of targets before fine-grained geolocation. This hypothesis is common to previous fine-grained methods. For example, SLG needs to estimate the coarse-grained location before estimating the fine-grained location of a target. And since SLG maps target IP addresses to landmarks, the largest estimated area is actually limited to the area that the known landmarks can cover. MLP-Geo also needs to cluster the landmarks and targets at first based on their measurement data, which is similar to coarse-grained IP geolocation. Corr-SLG also assumes that researchers already knew the city location of the targets and focus on finding fine-grained locations inside the city. This improved decoder can ease the burden of fine-tuning parameters by introducing some prior knowledge into GNN-based methods.

5.5 Model Training

To learn model parameters, we employ the Mean Square Error (MSE) loss between loc_{train} and $\hat{loc}_{train}$. In this work, we optimize the L_2 regularized MSE loss as follows:

$$\mathbf{Loss} = \sum_{i=1}^{N(v)_{train}} (loc_{train}^{ts} - \widehat{loc}_{train}^{ts})^2 + \lambda \|\Theta\|^2, \qquad (19)$$

where $N(v)_{train}$ denotes all nodes in the training dataset. And $\mathbf{\Theta}$ includes all the trainable model parameters of GNN-Geo. Deep learning methods usually suffer from overfitting. Besides Batch Normalization in the decoder, we also leverage L_2 regularization to prevent overfitting. Note that Batch Normalization is only used in training, and must be disabled during testing. λ controls the L_2 regularization strength. The Adam optimizer is employed to optimize the prediction model and update the model parameters by using the gradients of the loss function.

6 Performance Analysis

We evaluate experiments on 8 real-world datasets, aiming to answer the following research questions:

- RQ1: how does GNN-Geo perform as compared with state-of-the-art rule-based and learning-based street-level IP geolocation methods?
- RQ2: how does diffferent training-testing ratio affect GNN-Geo's performance?
- RQ3: how can rules help to improve GNN-Geo's vanilla MLP-based decoder?
- RQ4: how do different settings (e.g., basic GNN models, $f_{aggregation}$) affect the geolocating performance of GNN-Geo?

6.1 Dataset Description

To demonstrate the effectiveness and generalization capabilities of GNN-Geo, we plan to conduct experiments in various real-world computer networks. Due to privacy or intellectual property concerns, the datasets of most previous related works are not opened. So we need to collect our own datasets.

Target Areas and Landmarks. To test the generalization capabilities of GNN-Geo in different network environments, we collect different landmark datasets from 8 real-networks: (i) the IPv4 networks of New York State, Hong Kong, Shanghai, Beijing, and Tokyo; (ii) the IPv6 networks of Tokyo, Berlin and Munich. In section 6, they are referred as NYS, HK, Shanghai, Beijing, Tokyo, Tokyo-IPv6, Berlin-IPv6 and Munich-IPv6 datasets, respectively. These networks belong to three continents (North America, Asia and Europe), including both developed countries and developing countries. The internet service providers (ISP) in these areas are also different. For example, the main ISPs are Spectrum and Verizon in NYS; Asia Netcom in HK; China Telecom and China Unicom in Shanghai and Beijing. The urban environment, and terrain of these areas are also different. For example, NYS consists of several cities, including New York City, Buffalo city, Rochester city, etc. HK mainly consists of Hong Kong Island, Kowloon Peninsula, New Territories, and some smaller islands. Shanghai consists of the main urban area on the Changjiang alluvial plain as well as rural areas on a large island called Chongming Island. All 8 networks cover densely populated urban areas and sparsely populated rural areas. The sizes of these areas are also different as shown in Table 1.

We collect fine-grained IPv4 landmarks following the method in [5], which has been widely used by previous works. These landmarks are mainly famous organization website servers, whose IP addresses and organization locations are opened to the public. However, since more and more new website servers are deployed in the clouds, this collection method is not suitable for finding enough IPv6 fine-grained landmarks. Recently, Erik Rye and Robert Beverly [46] find many WiFi devices' IPv6 addresses and their geographical locations can be connected by their MAC address and WiFi BSSID (Basic Service Set Identifier). Based on this method, we collect IPv6 landmarks all over the world. In total, we find more than 350,000 IPv4/IPv6 landmarks. After verification and selection, there are 11,654 measurable fine-grained landmarks.

Each landmark is an IP address with a pair of (latitude, longitude). Due to limited space, we only show the landmark distribution in NYS, HK, and Shanghai in Fig. 3. The covering area and density of landmarks also could affect the performance of IP geolocation methods [47]. The number of landmarks in each network are shown in Table 1.

6.2 Engineering Applications of the Preprocessor and the Encoder

We deploy one probing host in each target network. The probing host is IPv4 (or IPv6) if the target network is IPv4-based (IPv6-based). Every probing host is responsible for getting the raw traceroute data in the area where it locates. The network graph of the area is then built based on the measurement data obtained by its probing host. For 10 days, one probing host probes all landmarks in its city every 10 minutes (1440 times in total). Each probing host is a server installed with network probing tools, of which function is similar to traceroute. The software can get the IP addresses of routers between the probing host and all landmarks. And it can also obtain the delays (accurate to milliseconds) between each router and the probing host. After the preprocessor, the number of nodes (IP addresses) and the number of the links are (2872, 3303) in NYS, (3071, 3697) in HK, (2530, 15108) in Shanghai, (3310, 18742) in Beijing, (1498, 4408) in Tokyo, (2297, 4966) in Tokyo-IPv6, (2070, 5776) in Berlin-IPv6 and (380, 856) in Munich-IPv6. So, their edge/node ratio is 1.15, 1.20, 5.97, 5.66, 2.94, 2.16, 2.79 and 2.25, respectively.

The initial attribute feature for each node is a 15-dimensional vector of values. The first 5 dimensions are the raw numerical of node delay and IP address. The numerical node delay is then binned into 10 intervals to form the last 10 dimensions. Here we use K-means to cluster delay into 10 bins. 10 is empirically selected. IP addresses can also be transformed into categorical features. The initial attribute feature for each edge is an 11-dimensional vector of values. The first dimension is the raw numerical data of the edge delay. The remaining 10 dimensions are the binned edge delay.

Actually, it is easy to combine more kinds of features into the proposed GNN-Geo, such as the domain name of routers, the two IP addresses of an edge, the delay changing trends of a node or edge, etc. All these features can be transformed into node/edge attribute features by the preprocessor, and then naturally introduced into model training through the encoder. And efforts can also be put into more sophisticated pre-processing techniques. For example, we can observe the histogram distribution of delay, and manually decide the number of bins for each area instead of using empirical numbers. IP addresses can also be explicitly transformed into categorical features. However, these features or preprocessing techniques are not easy to be applied in previous baselines for comparison. In this work, we focus on testing the learning potential of GNN in IP geolocation. Thus we utilize a simple preprocessor in this work, leaving further investigation on the preprocessor to future research.

6.3 Experiments Settings

Evaluation Metrics. In each network, we use 4 kinds of training/validation/testing ratios to compare the performance of GNN-Geo with baselines: training-70%, training-50%, training-30%, and training-10%. For all ratios, the validation ratio is 20%. We mainly change the ratio of training dataset and testing dataset: for training-70%, 70% landmarks are randomly chosen for training, 20% for validation, and 10% for testing; for training-50%, 50% landmarks are randomly chosen for training, 20% for validation, and 30% for testing; for training-30%, 30% landmarks are randomly chosen for training, 20% for validation, and 50% for testing; for training-10%, only 10% landmarks are randomly chosen for training, 20% for validation, and 70% for testing. When answering RQ1 (Section 6.5 and 6.6), RQ3 (Section 6.8) and RQ4 (Section 6.9), we mainly use train-70% datasets. Training-50%, training-30%, and training-10% are mainly used in answering RQ2 (Section 6.7).

We use the validation sets to do early stopping and hyper-parameters fine-tuning. We compare performances on the test sets if not specially explained. For each training set, the latitude and longitude ranges of landmarks are extended by 0.1 decimal degree, before being scaled to the $[0, 1]$. For example, assume the latitude and longitude range of one training set is (30.61N ~ 31.73N, 121.14E ~ 121.86E), then the range for scaling is (30.51N ~ 31.83N, 121.04E ~ 121.96E). 0.1 decimal degree is an empirical number, about 11.1 km for both longitude and latitude. We minimize the mean squared error loss following Formula 19. During the validation and test phase, we need to re-scale the output of the decoder into latitude and longitude values to calculate the geographical distance. The error distance of an IP address is the geographical distance between the ground-truth location and the estimated location. To evaluate the performance of IP geolocation, we adopt 2 evaluation metrics widely-used in previous works like [2, 5]: average distance error, and median distance error. We also leverage Cumulative Distribution Function (CDF) to show the distribution of error distances.

Parameter Settings. We implement GNN-Geo in Pytorch. Code and hashed datasets will be opened upon acceptance. The embedding sizes of the initial node ID feature in the encoder and the hidden node representation in the MP layers are searched between $\{32, 64, 128, 256\}$. We optimize the model with the Adam optimizer. We leverage the default initializer of Pytorch to initialize the model parameters and embeddings. For example, standard normal distribution $N(0, 1)$ is used to initialize node ID embeddings. A grid search is applied to find the best hyper-parameters [48]. The learning rate is tuned amongst $\{0.01, 0.001\}$, the coefficient of L_2 regularization is searched in $\{0.01, 0.001, 0.0005\}$. The depth of the MP layers L is searched in $\{1, 2, 3, 4, 5\}$. The number of hidden units of the edge network is searched in $\{4, 8, 16\}$. The dimension of initial edge embedding K is searched between $\{16, 8, 4\}$. The aggregation methods are searched among Mean, Sum and Max. Besides, the early stopping strategy is performed: training is stopped if the average distance error on the validation set does not increase for 1000 successive epochs. Usually, 4000 epochs are sufficient for GNN-Geo to converge.

6.4 Baselines

To evaluate the performance of GNN-Geo, we compare it with several state-of-the-art rule-based and learning-based street-level IP geolocation methods as follows.

SLG [5]. SLG is a typical rule-based IP geolocation method and is also one of the most widely-used street-level IP geolocation baselines. Its core rule is to map a target IP to the landmark which has the smallest relative delay to the target IP. There is no need to set or tune hyper-parameters for SLG. Thus in each area, we

TABLE 1: Performance (kilometers) Comparison of baselines and GNN-Geo

Areas	#Landmarks	Size(km^2)	Error Distance	Rule-based		Learning-based		%Error Reduced
				SLG [5]	Corr-SLG [2]	MLP-Geo [7]	GNN-Geo	
New York State	1705	141299	Average	38.176	38.303	43.274	**27.198**	28.76%
			Median	7.090	7.040	13.847	**4.901**	30.38%
Hong Kong	2061	1108	Average	13.019	10.001	9.793	**8.166**	16.61%
			Median	11.933	7.523	9.140	**6.596**	12.32%
Shanghai	1387	6340	Average	14.245	14.037	14.291	**10.368**	26.14%
			Median	11.056	*11.281*	11.659	**8.698**	21.33%
Beijing	1835	16410	Average	12.394	13.855	12.992	**9.523**	23.16%
			Median	10.259	11.314	10.985	**7.603**	25.89%
Tokyo	943	14034	Average	15.321	12.978	13.047	**12.306**	5.18%
			Median	13.206	10.217	9.701	**8.654**	10.79%
Tokyo (IPv6)	1949	14034	Average	15.372	7.066	3.912	**2.916**	25.46%
			Median	10.813	5.702	1.806	**1.344**	25.58%
Berlin (IPv6)	1503	883	Average	7.214	6.322	7.294	**3.368**	46.73%
			Median	5.141	5.100	6.300	**3.230**	36.67%
Munich (IPv6)	271	310	Average	7.908	5.028	4.270	**3.875**	9.25%
			Median	7.890	4.312	3.416	**3.230**	5.44%

[1] underline indicates the best metrics among baselines.

combine the training set and validation set to geolocate the testing set.

Corr-SLG [2]: Corr-SLG is a recent rule-based IP geolocation method. It divides landmarks into 1) Group A, strong-positive delay-distance-correlated landmarks; 2) Group B, strong-negative delay-distance-correlated landmarks; 3) Group C, weak delay-distance-correlated landmarks. The main hyper-parameters are C_a and C_b in Corr-SLG. They are used to divide all landmarks into Group A, Group B, and Group C based on delay-distance correlation. C_a and C_b are manually tuned among $\{0, 0.1, 0.2, \ldots, 0.9\}$ and $\{0.1, -0.2, \ldots, -0.9, -1\}$, respectively. We split the landmarks into a training set, a validation set and a testing set (same to GNN-Geo). In each area, we use the validation set to tune C_a and C_b , and then report the results on the test set. The parameter searching methods are the same as [2].

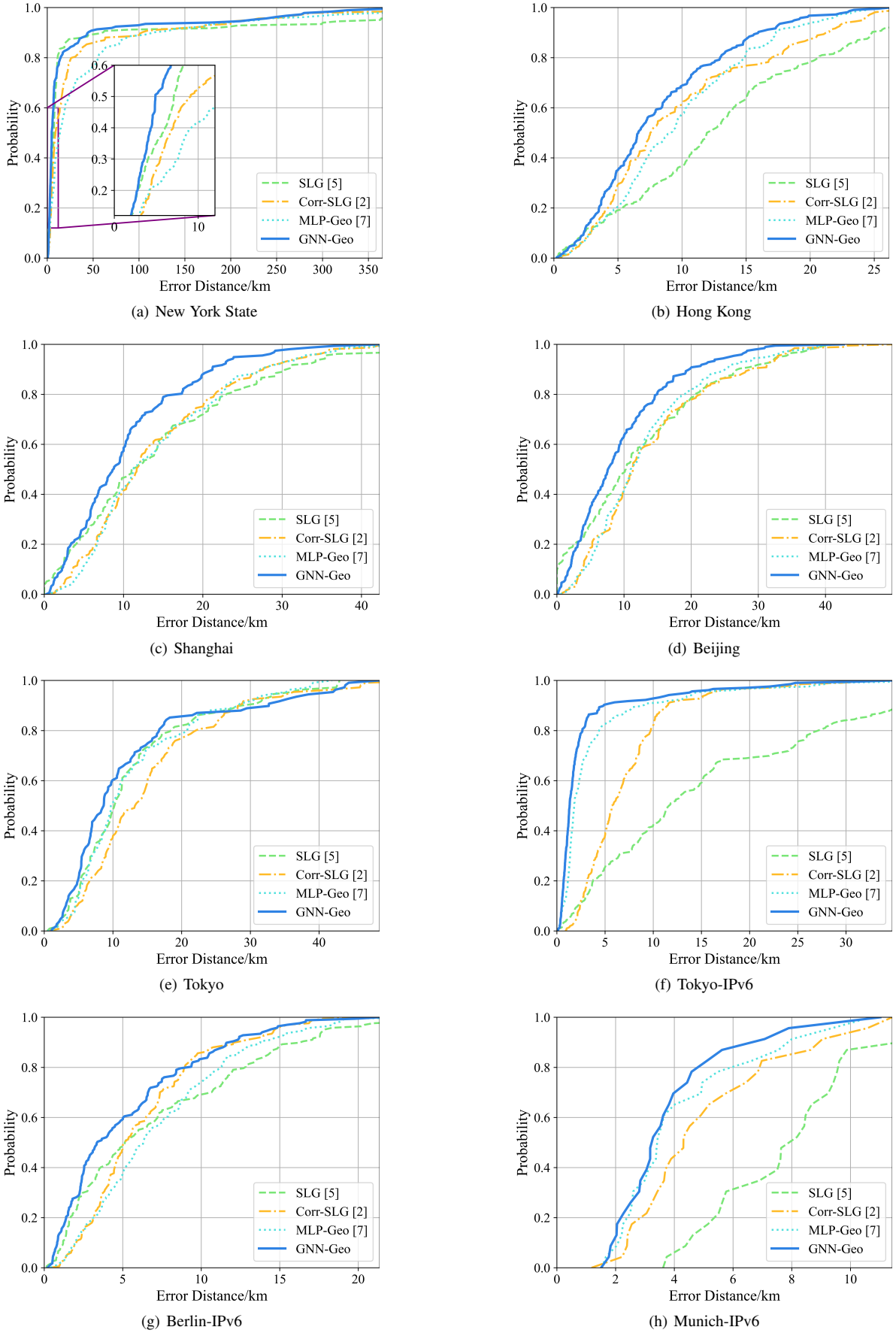
MLP-Geo [7]. MLP-Geo is a learning-based IP geolocation method. Like GNN-Geo and Corr-SLG, we use validation sets to tune the hyperparameters of MLP-Geo in each area. The main hyper-parameters of MLP-Geo include the learning rate, the number of training epochs and the dimension size of the middle dense layer. A grid search is applied to find the best hyper-parameters. The learning rate is tuned amongst $\{0.1, 0.01, 0.001, 0.0001\}$. The dimension size is searched in $\{32, 64, 128, 256\}$. The number of training epochs is 20,000 to find the epoch corresponding to the best performance. The other settings are kept the same as suggested in [7]. For example, we set its $beta$ to 30 when encoding the path.

6.5 Geolocation Performance Comparison between GNN-Geo and Baselines

The results of all methods on 8 datasets are shown in Table 1. The datasets are train-70% datasets. The numbers in Table 1 are averaged by 5 times of train-validation-testing (randomly changing seeds when splitting datasets). To achieve the best performance of all methods, parameter tuning of all methods is conducted as introduced in Section 6.4. From Table 1, **we can see that the proposed framework GNN-Geo consistently outperforms all baselines w.r.t. average error distance, median error distance and max error distance on all three datasets**. Please note that we leverage MSE Loss as an optimizer target function. MSE loss aims to minimize the whole loss on all data instances (not the median loss or the max loss). Thus we mainly discuss average error distance in this work. The best baseline (i.e., *italic* rows) in the following sections is the baseline with the smallest average error distance on the test sets if there is no special explanation.

Accuracy. From Table 1, on training-70% datasets, GNN-Geo outperforms the best baselines by 28.76%, 16.61%, 26.14%, 21.33%, 23.16%, 5.18%, 25.46%, 46.73%, and 9.25% in NYS (Corr-SLG), HK (MLP-Geo), Shanghai (Corr-SLG), Beijing (SLG), Tokyo (Corr-SLG), Tokyo-IPv6 (MLP-Geo), Berlin-IPv6 (Corr-SLG), and Munich-IPv6 (MLP-Geo) w.r.t. average error distance, respectively. And GNN-Geo also outperforms the best baselines by 30.38%, 12.32%, 21.33%, 25.89%, 10.79%, 25.58%, 36.67%, and 5.44% in NYS (Corr-SLG), HK (Corr-SLG), Shanghai (SLG), Beijing (SLG), Tokyo (MLP-Geo), Tokyo-IPv6 (MLP-Geo), Berlin-IPv6 (Corr-SLG), and Munich-IPv6 (MLP-Geo) w.r.t. median error distance, respectively. The best baselines are different in different networks and are shown in brackets. So averagely, in these 8 IPv4/IPv6 networks, GNN-Geo outperforms the best baselines by 22.51% and 21.05% w.r.t. average error distance and median error distance, respectively. These results demonstrate that the geolocation accuracy of GNN-Geo is clearly higher than baselines, which validates the value of investigating GNN's potential in fine-grained IP geolocation.

Generalization. **It is interesting that the best baselines are all different in 8 networks.** For average error distance, the best baseline is Corr-SLG for 4 times, MLP-Geo for 3 times, and SLG for 1 time; for median error distance, the best baseline is Corr-SLG for 3 times, MLP-Geo for 3 times, and SLG for 2 time. From the experiments, we can see the performances of baselines are not stable in different areas. Their performance varies significantly in different network environments. For SLG, it relies on the proportion of "strong-positive-delay-distance-correlated" landmarks. For Corr-SLG, the proportion of "strong-negative-delay-distance-correlated" landmarks decides whether it can outperform SLG. **Compared with previous baselines, GNN-Geo shows better generalization capabilities in different network environments**.



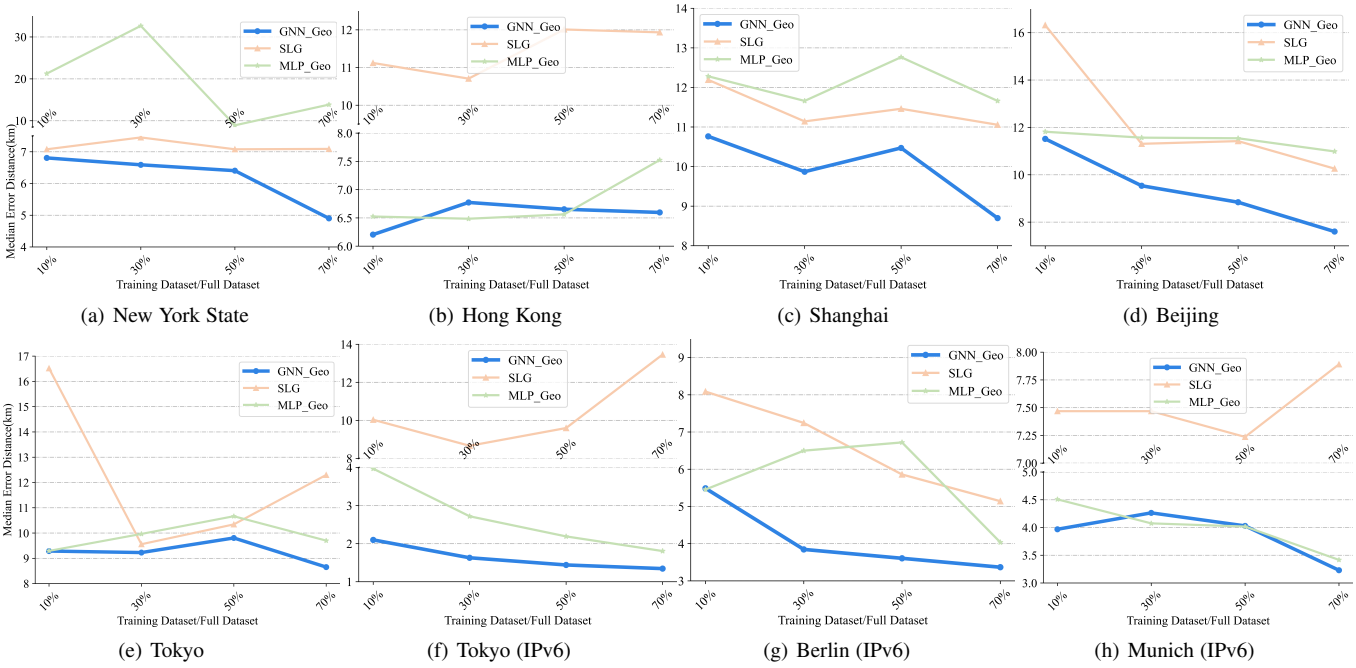
(a) New York State (b) Hong Kong

(c) Shanghai (d) Beijing

(e) Tokyo (f) Tokyo-IPv6

(g) Berlin-IPv6 (h) Munich-IPv6

Fig. 4: Error Distance CDF of 8 Networks (the maximum values of x-axis are GNN-Geo's max error distances)



(a) New York State (b) Hong Kong (c) Shanghai (d) Beijing

(e) Tokyo (f) Tokyo (IPv6) (g) Berlin (IPv6) (h) Munich (IPv6)

Fig. 5: The Relationship between Error Distance and Training dataset/Full dataset

6.6 An Interesting Difference between Learning-based and Rule-based Methods

Fig. 4 shows the cumulative probability of error distances of baselines and GNN-Geo in all 8 networks (training-70% datasets). In NYS network, the max error distances are too large, overlapping the smaller cumulative probability with each other. Thus we magnify the picture (0 - 10 km & 0.1 - 0.6) in Fig. 4(a).

From Fig. 4, we can see that: generally, the error distances of GNN-Geo are smaller than baselines in most phases of cumulative probability. This validates GNN-Geo's advantages in most cases. However, if we check the error distances less than 5 km, we can see SLG sometimes could be better than GNN-Geo in the very beginning. For example, in Shanghai and Beijing, SLG is clearly better than GNN-Geo in the smallest error distances (<2km). In fact, some smallest error distances of SLG are even 0 km. Why SLG could be better than GNN-Geo at the start and become worse in the latter part of cumulative probability?

The rule of SLG is: "the shortest delay comes from the smallest distance". Thus SLG could get very accurate geolocation results for targets that follow this rule. Some error distances could be close to 0 if the landmarks are at the same location as the targets. This is why SLG performs well in the first part. However, the disadvantage of SLG is that its error distances could be very large for targets that follow the opposite rule – "the shortest delay comes from the longest distance". Besides, the error distances of SLG are also large for targets which have no clear linear delay-distance relationship at all. This is why the max error distances of SLG are very large in some datasets. If we observe the max error distances when probability reaches 100%, we can see SLG's max error distances are much larger than other baselines. For example, in Tokyo-IPv6, we could see when GNN-Geo's max error distance is just about SLG's top 85% max error distance. So the average error distances of SLG are not so good if there are many of these kinds of targets. The cumulative probability of SLG would quickly become worse in the latter part.

GNN-Geo does not follow a specific linear delay-distance rule. Its "measurement data - location" mapping function is learned based on a supervised-learning method. The mapping function is optimized by minimizing the MSE loss of the whole training set. In other words, GNN-Geo considers the whole performance or average performance of all targets. Its mapping function tries to be suitable to as many targets as possible, instead of only focusing on a part of targets. In a dataset, if different targets follow different delay-distance or topology-distance rules, the mapping function of GNN-Geo may be less accurate for some targets than SLG. This is because the mapping function needs to get better performance for other targets which do not follow linear rules. Therefore GNN-Geo can outperform SLG in the latter part of cumulative probability. Actually, we can notice that MLP-Geo follows a similar trend with GNN-Geo. In Fig. 4(c), MLP-Geo is worse than SLG before 3 km and quickly becomes better afterward. This is because MLP-Geo is also a learning-based method and is optimized by MSE loss.

6.7 How training ratios affect GNN-Geo's Performance

Fig. 5 shows how different methods' performances are affected by changing training-validation-testing ratios in all datasets. Here, we select a typical method in rule-based learning (SLG) and learning-based learning methods (MLP-Geo) to compare with GNN-Geo. Parameters are re-tuned for learning-based methods when changing ratios. We notice that, though average error distances generally follow similar trends with median error distances, average error distances are easier to be affected by max error distance. Since the training ratio become smaller, some methods may have too large average/max error distances, which make other results in the same figures hard to distinguish. So, we mainly use median error distance to show the trends in Fig. 5.

From Fig. 5, we can see that the performances of all methods generally become better when training ratio increases from 10% to 70%, validating that more landmarks in the same network usually lead to better geolocation accuracy for measurement-based

TABLE 2: Performance (kilometers) Comparison of Different Decoder Types

Decoder Type		New York State		Hong Kong		Shanghai	
		Average	Epochs	Average	Epochs	Average	Epochs
Pure MLP-based Decoders	Vanilla MLP	3406.247	512	2690.566	10000	3266.079	10000
	+BatchNorm	6010.929	26	2812.680	385	1087.136	20
MLP-based Decoders with Rules	Vanilla MLP	69.663	6643	8.127	9828	12.504	3589
	+BatchNorm	29.861	2073	7.735	4690	9.723	1808
	+Sigmoid	26.445	1484	7.681	2265	9.843	1096
	+BatchNorm+Sigmoid	**25.165**	**3534**	**7.603**	**760**	**9.673**	**1439**

[1] **Bold** indicate the average error distance of GNN-Geo, underline rows indicate the validation average error distances among other decoder types. Note all these methods only differ in decoder types.
[2] "Epochs" indicates the training epoch number of the corresponding results.
[3] These results are the best average distance errors on validation sets.

methods. In 18 out of 24 cases (75%), GNN-Geo still clearly outperforms baselines with training ratio less than 70%. This shows that we still can use GNN-Geo even with relatively less landmarks. However, if the training ratio is too less (training-ratio is 10%), only in 50% cases, GNN-Geo is clearly better than other baselines. This shows that, compared with other methods, GNN's performances may be more easily affected with extremely smaller training datasets. This is because: (1) compared with SLG, as a learning-based method, GNN-Geo could be overfit on extremely sparse datasets; (2) compared with MLP-Geo, GNN-Geo has more parameters to train, so it is generally easier to overfit. So sometimes, if the training dataset is too small, and the signals (e.g., network topology) extracted by GNN-Geo are not so important to IP geolocation, then GNN-Geo's advantages could be decreased.

6.8 Geolocation Performance Comparison over different Decoders

As explained in section 5.4, GNN-based geolocation methods with the original decoders may be not suitable for IP geolocation. Here we compare the proposed decoder with several baseline decoders. All these decoders share the same preprocessor, encoder and MP layers. Pure MLP-based decoders do not limit the output data range. MLP-based decoders improved by rules scale the latitude and longitude range of the training set (+11 km) to [0,1]. For **Vanilla MLP**, the decoder consists of a dense layer with Relu and an output layer. For **+BatchNorm**, the decoder consists of a dense layer with Relu, a batch normalization layer and an output layer. For **+Sigmoid**, the decoder consists of a dense layer with Relu and an output layer with Sigmoid. For **+BatchNorm+Sigmoid**, the decoder consists of a dense layer with Relu, a batch normalization layer and an output layer with Sigmoid. **+BatchNorm+Sigmoid** is the proposed rule-based decoder used in GNN-Geo.

The best geolocation performance w.r.t average error distance of all decoders on validation sets is shown in Table 2. Here we only show the results in first 3 networks (NYS, HK, Shanghai) with training-70% ratio as an example. For each dataset, the **bold** indicates the best average error distance for all decoders while the *italic* row indicates the best average error distance for all baseline decoders. "Epochs" indicate the training epoch number of the corresponding results. It shows how many epochs a decoder needs to converge. The training epochs are at most 10000 and early stopped if results are not improved in 1000 epochs.

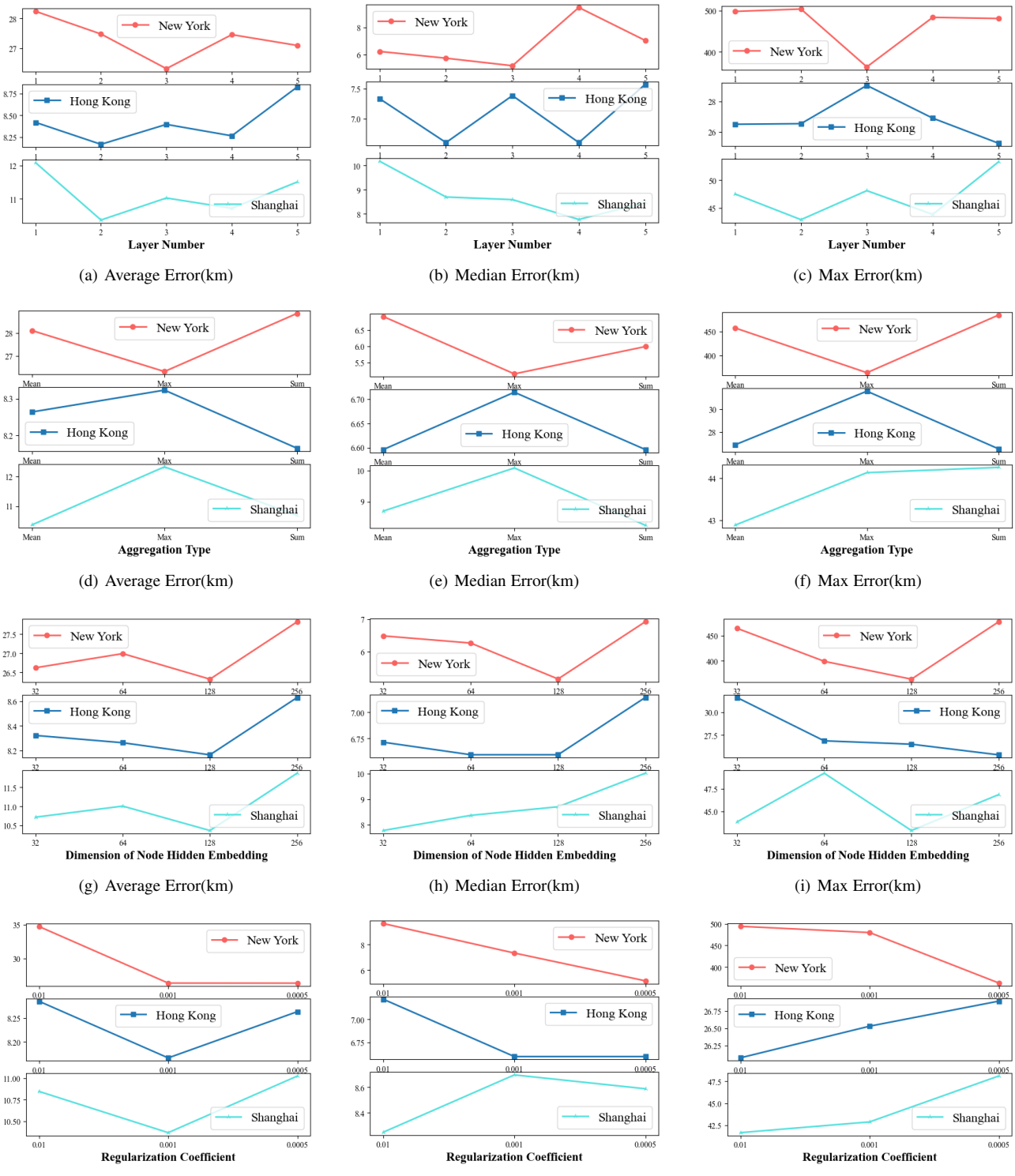
From Table 2, we can see the best average distance errors of two pure MLP-based decoders are more than 1,000 km even after 10,000 training epochs. This is because most of the output data belong to $\mathbb{R}$ and are far away from the real locations of the targets. **It indicates that GNN is not a "silver bullet" to the IP geolocation problem**. From Table 1, we can the average distances of GNN-Geo are between 8 km and 28 km, clearly outperforming all baselines. However, simply changing its decoder could lead to unacceptable performance. **The original GNN cannot be directly used for IP geolocation**. We still need to carefully design the structure of the whole GNN-Geo framework to gain better geolocation performance.

The performances of decoders improved with rules are much better. We can find some trends: 1) **+Sigmoid** and **+BatchNorm** both can clearly increase the performance and significantly reduce the epoch numbers for convergence; 2) **+Sigmoid** is more accurate than **+BatchNorm** except Shanghai dataset; 3) **+Sigmoid** converges faster than **+BatchNorm**; 4) **+BatchNorm+Sigmoid** is better than only using **+BatchNorm** or **+Sigmoid**. We explain these trends as follows. Sigmoid limits the range of output into $[0, 1]$, which eases the burden for the optimizer. All the output data must be in the area of the training datasets, which is a coarse-grained location estimation of the targets. However, only using Sigmoid could meet the saturating problem. Batch normalization can help to relieve this problem. Batch normalization itself can also facilitate neural network training [45, 49]. Besides, it is noticed that rule-based Vanilla MLP is much better than the original Vanilla MLP, with its output still belonging to $\mathbb{R}$. This is because our initializers sometimes would make the decoder's estimated data in the first several epochs close to 0. If this happens, its performance could be much better than the original MLP. These findings show that we can combine the advantages of rule-based and learning-based methods together to gain better performance.

These findings and the advantages of SLG shown in Section 6.6 indicate that we should consider combining the advantages of rule-based and learning-based methods together to gain better performance.

6.9 Further Investigation on GNN-Geo Properties

In this section, we investigate the impact of different learning configurations on the performance of GNN-Geo. We start by exploring the influence of the depth of the MP layers. We then study how different message aggregation methods affect the performance. The effect of Regularization Coefficients is discussed later. Then, we analyze the influences of Graph Node Embedding Size. In Fig. 6, we show the effect of parameters on average error distance, median error distance and max error distance with training-70% datasets in the first 3 networks (NYS, HK and Shanghai) as an example. However, we mainly analyze the effect of parameters on average error distance because the



(a) Average Error(km) (b) Median Error(km) (c) Max Error(km)

(d) Average Error(km) (e) Median Error(km) (f) Max Error(km)

(g) Average Error(km) (h) Median Error(km) (i) Max Error(km)

(j) Average Error(km) (k) Median Error(km) (l) Max Error(km)

Fig. 6: Parameter Study of GNN-Geo in New York State, Hong Kong, Shanghai

optimized target is MSE loss. GNN-Geo did not specifically seek a minimum median error distance or max error distance. We pick the best performance for each parameter on the testing sets. And no matter how parameters vary, the average error distances in Fig. 6 are consistently smaller than baselines in previous works for all datasets. It verifies the effectiveness of GNN-Geo, empirically showing that explicitly modeling the network connectivity can greatly facilitate the geolocation task. Finally, we also investigate how different GNN basic models would affect the performance of GNN-Geo.

6.9.1 Effect of the Depth of Message Passing Layers

The depth of the MP layers is actually the number of MP layers. From Fig. 6(a), the best layer numbers are 3, 2, 2 in NYS, HK and Shanghai, reflectively. The worst layer numbers are 1, 5 and 1 in NYS, HK and Shanghai, respectively. Compared to the worst layer numbers, the improvements of the best layer numbers are 7.27% ~ 16.55%. Increasing the depth of GNN-Geo substantially enhances the geolocation performance. GNN-Geo-3 indicates GNN-Geo with three message propagation layers and similar notations for others. We can see that GNN-Geo-2, GNN-Geo-3 and GNN-Geo-4 achieve consistent improvement over GNN-Geo-1 across all datasets, which only consider the first-order neighbors only. When further stacking the propagation layer on the top of GNN-Geo-4, we find that GNN-Geo-5 leads to overfitting on the Hong Kong dataset. This might be caused by applying a too deep architecture that might introduce noises to the representation learning. The marginal improvements on the other two datasets verify that conducting 2-3 propagation layers are sufficient to model the network for IP geolocation.

6.9.2 Effect of Aggregation Methods

From Fig. 6(d), the best aggregation methods are Max, Mean, Mean in NYS, HK and Shanghai, respectively. The worst aggregation methods are Sum, Max, Max in NYS, HK and Shanghai, respectively. Compared to the worst aggregation methods, the improvements of the best aggregation methods are 9.58%, 1.92%, 18.95%, respectively. Neighborhood aggregation is a key operation, which distinguishes GNN from MLP as we discussed in section 4. Theoretically, all three methods have their limitation: 1) both Mean and Sum cannot treat each neighbor differently; 2) Max may only lose some neighbors' useful information. Previous work [44] has pointed out that Sum may be more powerful than Mean and Max for graphs without attributes. However, in three datasets of this paper, Mean is the best or the second-best, while Max is a little worse than Mean. Max could be the worst or the best in different datasets. The attributed graph is more complex because the attributes of edges and nodes are also influencing the performance besides the network structure. The best aggregation methods in attributed graphs for geolocation still need more depth analysis.

6.9.3 Effect of Graph Node Embedding Size

From Fig. 6(g), all the best graph node embedding sizes are 128 in NYS, HK and Shanghai. All the worst graph node embedding sizes are 256 in NYS, HK and Shanghai. Compared to the worst graph node embedding sizes, the improvements of the best graph node embedding sizes are 5.71%, 5.68%, 14.49%, respectively. The influence patterns of Graph Node Embedding Size are stable for all three datasets. Increasing embedding size brings more representation power to the MP layers by introducing more complexity. This enables the model to learn more complicated patterns. So embedding size 128 performs better than 32 and 64 in all datasets. However, a too large embedding size like 256 could disturb the model with noises and result in overfitting, especially if there are not enough data instances for training.

6.9.4 Effect of Regularization Coefficients

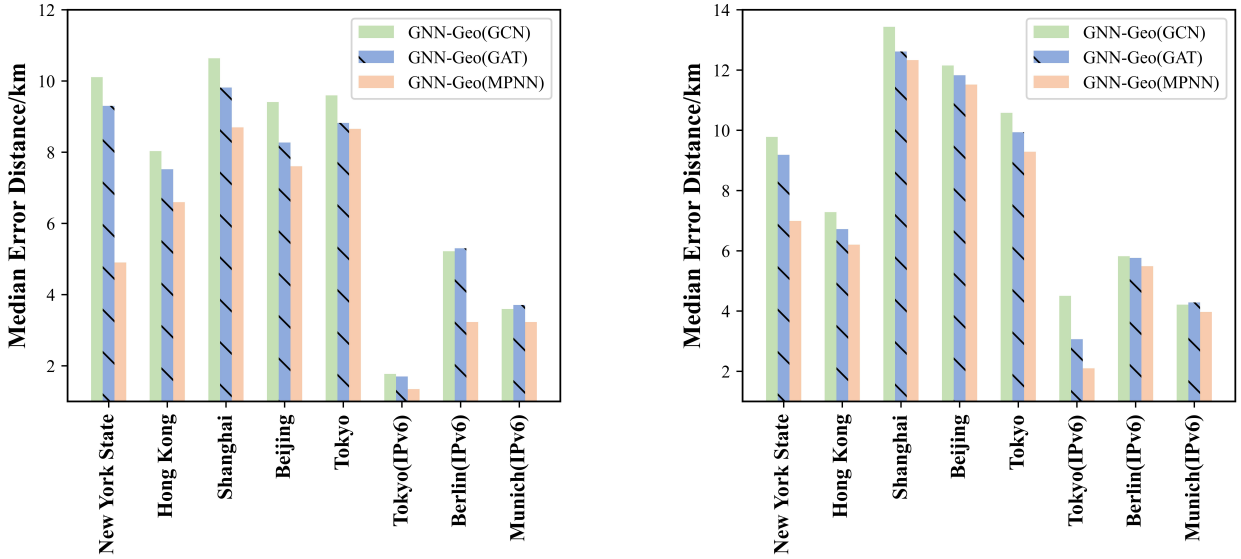
From Fig. 6(j), the best Regularization Coefficients are 0.0005, 0.001, 0.001 in NYS, HK and Shanghai, respectively. The worst Regularization Coefficients are 0.01, 0.01, 0.0005 in NYS, HK and Shanghai, respectively. Compared to the worst Regularization Coefficients, the improvements of the best Regularization Coefficients are 31.90%, 1.47%, 6.36%. L_2 Regularization is the main method in GNN-Geo to overcome overfitting. However, too larger Regularization Coefficients could lead to underfitting. We find that the influence of Regularization Coefficients varies significantly in different datasets. The Regularization Coefficient is very important to NYS dataset since the improvement is larger than all the other parameters discussed in this section. However, the Regularization coefficient is the least influential parameter in the Hong Kong dataset, compared with the other parameters. Generally, the Regularization Coefficients 0.001 should be tried first because it is the best and second-best in all datasets.

6.9.5 Effect of Regularization Coefficients

From Fig. 6(j), the best Regularization Coefficients are 0.0005, 0.001, 0.001 in NYS, HK and Shanghai, respectively. The worst Regularization Coefficients are 0.01, 0.01, 0.0005 in NYS, HK and Shanghai, respectively. Compared to the worst Regularization Coefficients, the improvements of the best Regularization Coefficients are 31.90%, 1.47%, 6.36%. L_2 Regularization is the main method in GNN-Geo to overcome overfitting. However, too larger Regularization Coefficients could lead to underfitting. We find that the influence of Regularization Coefficients varies significantly in different datasets. The Regularization Coefficient is very important to NYS dataset since the improvement is larger than all the other parameters discussed in this section. However, the Regularization coefficient is the least influential parameter in the Hong Kong dataset, compared with the other parameters. Generally, the Regularization Coefficients 0.001 should be tried first because it is the best and second-best in all datasets.

6.9.6 Effect of Different basic GNN models

In section 5 and all previous experiments of Section 6, we select MPNN as the basic GNN model for message passing layers. However, besides MPNN, there are still other famous basic GNN models such as GCN [11] and GAT [23]. Can they be better than MPNN for IP geolocation tasks? To answer this question, GCN and GAT are also used for message passing layers, which are referred as GNN-Geo-GCN and GNN-Geo-GAT, respectively. We select training-70% and training-10% to test their performances. After parameters tuning, the average error distances and median error distances of GNN-Geo (GCN), GNN-Geo (GAT) and GNN-Geo (MPNN) are shown in Fig. 7. From Fig. 7, we can see for both average and median error distances, GNN-Geo (MPNN) outperforms the other two methods on training-70% and training-10% datasets. This validates the importance to explicitly model edge features for IP geolocation. GNN-Geo (GAT) is usually better than GNN-Geo (GCN), which may indicate that some neighbors should be more important than others, and attention mechanism should also be tried in improving GNN-Geo in the future. Besides,



(a) Training Dataset/Full Dataset = 70%

(b) Training Dataset/Full Dataset = 10%

Fig. 7: Median Error Distances of Different GNN Basic Models in 8 Networks

compared with train-10%, the advantages of MPNN are usually clearer in training-70% datasets. This may show that GNN-Geo (MPNN) is easier to overfit than other two methods on smaller training datasets.

7 Discussion

In this paper, we mainly test whether GNN-Geo's accuracy outperforms previous fine-grained IP geolocation methods. From the experiments in Section 6.5, we can see that GNN-Geo is more accurate than previous methods due to its stronger modeling ability on non-linear relationships hidden in computer network measurement data. In this section, we will discuss what improvements GNN-Geo still needs to make it more practical for industrial-scale IP geolocation tasks.

Increasing Accuracy with fewer Landmarks. GNN is a deep learning-based method. Like other deep learning-based methods, its performance is much easier to be affected than rule-based methods when the landmarks or training data instances are too few (as shown in Section 6.7). This is because deep learning methods are easier to overfit on sparse datasets. Thus, we need to consider how to improve GNN-Geo's performance in areas where the landmarks are hard to collect.

Injecting explainability into GNN-Geo Compared with rule-based methods like SLG, deep learning-based methods often lack explainability for results. For example, for a target IP address in the Shanghai dataset, after geolocating it with SLG, it is easier to analyze why the error distance of this target is large. It is usually because its shortest delay does not come from its nearest landmark. However, it is hard to explain why GNN-Geo's error distance for this target is large. We need to discuss how to inject explainability into GNN-Geo's results for IP geolocation service users.

8 Conclusion

In this work, we first formulate the measurement-based fine-grained IP geolocation task into a semi-supervised attributed-graph node regression problem. Subsequently, we discuss why GNN is worth to be tested for IP geolocation. Then, we propose a GNN-based fine-grained IP geolocation framework (GNN-Geo) to solve the node regression problem. The framework consists of a preprocessor, an encoder, MP layers and a decoder. Moreover, we ease the convergence problem of GNN-based IP geolocation methods by scaling the range of output as well as combining the batch normalization and Sigmoid functions into the decoder. Finally, the experiments in 8 different real-world IPv4/IPv6 datasets verify the generalization capabilities of GNN-Geo for fine-grained IP geolocation.

The main advantages of GNN-Geo are the strong modeling ability on the computer network measurement data and the powerful extraction ability of non-linear relationships. Compared with rule-based methods, its main disadvantage is explainability, which is also a common problem for deep learning-based methods. Compared with the previous deep learning-based methods, the stronger modeling ability of GNN-Geo also needs more computing resources for training. In future work, we plan to increase GNN-Geo's accuracy with fewer landmarks, inject explainability into GNN-Geo, lower the training cost of GNN-Geo and improve GNN-Geo with various mechanisms like attention.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant No. 2022YFB3102904), the National Natural Science Foundation of China (Grant No.U1804263, 61872448 and 62172435), and Zhongyuan Science and Technology Innovation Leading Talent Project of China (Grant No. 214200510019).

References

[1] Q. Li, Z. Wang, D. Tan, J. Song, H. Wang, L. Sun, and J. Liu, "Geocam: An ip-based geolocation service through fine-grained and stable webcam landmarks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1798–1812, 2021.

[2] S. Ding, F. Zhao, and X. Luo, "A street-level ip geolocation method based on delay-distance correlation and multilayered common routers," *Security and Communication Networks*, vol. 2021, no. 1, pp. 1–10, 2021.

[3] Z. Wang, Q. Li, J. Song, H. Wang, and L. Sun, "Towards ip-based geolocation via fine-grained and stable webcam landmarks," in *Proceedings of The Web Conference (WWW '20)*, pp. 1422–1432, 2020.

[4] O. Dan, V. Parikh, and B. D. Davison, "Ip geolocation through reverse dns," *ACM Transaction on Internet Technology*, vol. 22, no. 1, pp. 17:1–17:29, 2022.

[5] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent ip geolocation.," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11)*, vol. 11, pp. 27–27, 2011.

[6] H. Liu, Y. Zhang, Y. Zhou, D. Zhang, X. Fu, and K. Ramakrishnan, "Mining checkins from location-sharing services for client-independent ip geolocation," in *IEEE Conference on Computer Communications (INFOCOM '14)*, pp. 619–627, 2014.

[7] F. Zhang, F. Liu, and X. Luo, "Geolocation of covert communication entity on the internet for post-steganalysis," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–10, 2020.

[8] H. Jiang, Y. Liu, and J. N. Matthews, "Ip geolocation estimation using neural networks with stable landmarks," in *IEEE Conference on Computer Communications Workshops (INFOCOM '16 WKSHPS)*, pp. 170–175, 2016.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[10] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

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

[12] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *International ACM Conference on Research and Development in Information Retrieval (SIGIR '21)*, pp. 726–735, 2021.

[13] Y. Wu, D. Lian, S. Jin, and E. Chen, "Graph convolutional networks on user mobility heterogeneous graphs for social relationship inference," in *International Joint Conference on Artificial Intelligence (IJCAI '19)*, pp. 3898–3904, 2019.

[14] J. Wu, W. Shi, X. Cao, J. Chen, W. Lei, F. Zhang, W. Wu, and X. He, "Disenkgat: Knowledge graph embedding with disentangled graph attention network," in *ACM International Conference on Information & Knowledge Management (CIKM '21)*, pp. 2140–2149, 2021.

[15] O. Dan, V. Parikh, and B. D. Davison, "Ip geolocation using traceroute location propagation and ip range location interpolation," in *Companion Proceedings of the Web Conference (WWW '21)*, pp. 332–338, 2021.

[16] P. Kumar, R. Kumar, G. Srivastava, G. P. Gupta, R. Tripathi, T. R. Gadekallu, and N. N. Xiong, "Ppsf: A privacy-preserving and secure framework using blockchain-based machine-learning for iot-driven smart cities," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2326–2341, 2021.

[17] J. Guo, A. Liu, K. Ota, M. Dong, X. Deng, and N. N. Xiong, "Itcn: An intelligent trust collaboration network system in iot," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 203–218, 2022.

[18] X. Chen, Z. Zhang, A. Qiu, Z. Xia, and N. N. Xiong, "Novel coverless steganography method based on image selection and stargan," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 219–230, 2022.

[19] Z. Xia, L. Wang, J. Tang, N. N. Xiong, and J. Weng, "A privacy-preserving image retrieval scheme using secure local binary pattern in cloud computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 318–330, 2021.

[20] Y. Mao, L. Zhou, and N. Xiong, "TPS: A topological potential scheme to predict influential network nodes for intelligent communication in social networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 529–540, 2021.

[21] N. Nagaraj, H. L. Gururaj, B. H. Swathi, and Y.-C. Hu, "Passenger flow prediction in bus transportation system using deep learning," *Multimedia tools and applications*, vol. 81, no. 9, pp. 12519–12542, 2022.

[22] W. Jiang, "Graph-based deep learning for communication networks: A survey," *arXiv: Networking and Internet Architecture*, 2021.

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

[24] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning (ICML '17)*, pp. 1263–1272, 2017.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[26] P. Chen, W. Liu, C.-Y. Hsieh, G. Chen, and S. Zhang, "Utilizing edge features in graph neural networks via variational information maximization," *arXiv preprint arXiv:1906.05488*, 2019.

[27] F. Zhou, T. Wang, T. Zhong, and G. Trajcevski, "Identifying user geolocation with hierarchical graph neural networks and explainable fusion," *Information Fusion*, vol. 81, pp. 1–13, 2022.

[28] Q. Xuan, K. Qiu, J. Zhou, Z. Chen, D. Xu, S. Zheng, and X. Yang, "Adaptive visibility graph neural network and its application in modulation classification," *CoRR*, vol. abs/2106.08564, 2021.

[29] Y. Song, H. Ye, M. Li, and F. Cao, "Deep multi-graph neural networks with attention fusion for recommendation," *Expert Systems with Applications*, vol. 191, p. 116240, 2022.

[30] X. Zhang, F. Lin, D. Dong, W. Chen, and B. Liu, "Heterogeneous graph attention network for user geolocation," in *Pacific Rim International Conference on Artificial Intelligence*, pp. 433–447, Springer, 2021.

[31] F. M. Funes, J. I. Alvarez-Hamelin, and M. G. Beiró, "Designing weighted and multiplex networks for deep learning user geolocation in twitter," *arXiv preprint arXiv:2112.06999*, 2021.

[32] W. Jing, X. Song, D. Di, and H. Song, "geogat: Graph model based on attention mechanism for geographic text classification," *Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, pp. 1–18, 2021.

[33] A. Badia-Sampera, J. Suárez-Varela, P. Almasan, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Towards more realistic network models based on graph neural networks," in *International Conference on emerging Networking EXperiments and Technologies (CoNEXT '19)*, pp. 14–16, 2019.

[34] M. F. Galmes, J. Suárez-Varela, P. Barlet-Ros, and A. Cabellos-Aparicio, "Applying graph-based deep learning to realistic network scenarios," *arXiv: Networking and Internet Architecture*, 2020.

[35] F. Geyer and S. Bondorf, "Deeptma: Predicting effective contention models for network calculus using graph neural networks," in *IEEE Conference on Computer Communications (INFOCOM '19)*, pp. 1009–1017, 2019.

[36] F. Geyer and S. Bondorf, "Graph-based deep learning for fast and tight network calculus analyses," *IEEE Transactions on Network Science and Engineering*, vol. 8, pp. 75–88, 2021.

[37] K. Rusek and P. Chołda, "Message-passing neural networks learn little's law," *IEEE Communications Letters*, vol. 23, no. 2, pp. 274–277, 2018.

[38] T. Mallick, M. Kiran, B. Mohammed, and P. Balaprakash, "Dynamic graph neural network for traffic forecasting in wide area networks," in *IEEE International Conference on Big Data (Big Data '20)*, 2020.

[39] C. Yang, Z. Zhou, H. Wen, and L. Zhou, "Mstnn: A graph learning based method for the origin-destination traffic prediction," in *International Conference on Communications (ICC '20)*, pp. 1–6, 2020.

[40] Z. Jianlong, H. Qu, J. Zhao, H. Dai, and D. Jiang, "Spatiotemporal graph convolutional recurrent networks for traffic matrix prediction," in *Transactions on Emerging Telecommunications Technologies*, 2020.

[41] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, "Automating botnet detection with graph neural networks.," *arXiv: Cryptography and Security*, 2020.

[42] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[43] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *International Conference on Neural Information Processing Systems (NIPS '17)*, pp. 1025–1035, 2017.

[44] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.

[45] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning (ICML '15)*, pp. 448–456, 2015.

[46] E. Rye and R. Beverly, "Ipvseeyou: Exploiting leaked identifiers in ipv6 for street-level geolocation," *arXiv preprint arXiv:2208.06767*, 2022.

[47] G. Ciavarrini, M. S. Greco, and A. Vecchio, "Geolocation of internet hosts: Accuracy limits through cramér–rao lower bound," *Computer Networks*, vol. 135, pp. 70–80, 2018.

[48] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[49] S. Santurkar, D. Tsipras, A. Ilyas, and A. Mądry, "How does batch normalization help optimization?," in *International conference on neural information processing systems (NIPS '18)*, pp. 2488–2498, 2018.

Shichang Ding received the Ph.D. degree from University of Göttingen, Germany, in 2020. He is currently a lecturer at State Key Laboratory of Mathematical Engineering and Advanced Computing. His research interests include cyberspace surveying and mapping, graph deep learning, and social computing.

Xiangyang Luo received the Ph.D. degree from Information Engineering University, China, in 2010. He is currently a Professor and a Ph.D. Supervisor with State Key Laboratory of Mathematical Engineering and Advanced Computing. He has authored or coauthored more than 150 refereed international journal and conference papers. His research interests include multimedia security and cyberspace surveying and mapping.

Jinwei Wang received the Ph.D. degree in information security from the Nanjing University of Science and Technology, China, in 2007. He is currently a Professor at the Nanjing University of Information Science and Technology. He has published over 50 papers. His research interests include multimedia copyright protection, multimedia forensics, multimedia encryption, and data authentication.

Xiaoming Fu received the PhD degree in computer science from Tsinghua University, Beijing, China in 2000. He was then a research staff at Technical University of Berlin until joining the University of Göttingen, Germany in 2002, where he has been a professor in computer science and heading the computer networks group since 2007. He has spent research visits at Cambridge, Columbia, UCLA, Tsinghua University, Uppsala, and UPMC, and is an IEEE fellow and distinguished lecturer. His research interests include Internet-based systems, applications, and social networks. He is currently an editorial board member of IEEE Communications Magazine, IEEE Transactions on Network and Service Management, Elsevier Computer Networks, and Computer Communications, and has published over 150 peer-reviewed papers in renowned journals and international conference proceedings.