Connecting the Hosts: Street-Level IP Geolocation with Graph Neural Networks

Zhiyuan Wang*
University of Electronic Science and Technology of China
zhy.wangcs@gmail.com

Fan Zhou*†
University of Electronic Science and Technology of China
fan.zhou@uestc.edu.cn

Wenxuan Zeng*
University of Electronic Science and Technology of China
zwx.andy@outlook.com

Goce Trajcevski
Iowa State University
gocet25@iastate.edu

Chunjing Xiao
Henan University
chunjingxiao@gmail.com

Yong Wang
Hong Kong University of Science and Technology
wangyongjoy@ust.hk

Kai Chen
Hong Kong University of Science and Technology
kaichen@ust.hk

ABSTRACT

Pinpointing the geographic location of an IP address is important for a range of location-aware applications spanning from targeted advertising to fraud prevention. The majority of traditional measurement-based and recent learning-based methods either focus on the efficient employment of topology or utilize data mining to find clues of the target IP in publicly available sources. Motivated by the limitations in existing works, we propose a novel framework named GraphGeo, which provides a complete processing methodology for street-level IP geolocation with the application of graph neural networks. It incorporates IP hosts knowledge and kinds of neighborhood relationships into the graph to infer spatial topology for high-quality geolocation prediction. We explicitly consider and alleviate the negative impact of uncertainty caused by network jitter and congestion, which are pervasive in complicated network environments. Extensive evaluations across three large-scale real-world datasets demonstrate that GraphGeo significantly reduces the geolocation errors compared to the state-of-the-art methods. Moreover, the proposed framework has been deployed on the web platform as an online service for 6 months.

CCS CONCEPTS

• **Networks** → **Location based services**; • **Computing methodologies** → *Spatial and physical reasoning*.

*Equal contribution.
†Corresponding author: Fan Zhou.

KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539049

KEYWORDS

IP Geolocation; Network Topology; Graph Neural Networks; Uncertain Inference; Ubiquitous Computing

ACM Reference Format:
Zhiyuan Wang*, Fan Zhou*†, Wenxuan Zeng*, Goce Trajcevski, Chunjing Xiao, Yong Wang, and Kai Chen. 2022. Connecting the Hosts: Street-Level IP Geolocation with Graph Neural Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, Washington, DC, USA, 11 pages. https://doi.org/10.1145/3534678.3539049

1 INTRODUCTION

An IP address is a unique identifier assigned to each host connected to the Internet, which serves as personally identifiable information and virtual location addressing. IP geolocation refers to the process of allocating a real-world geographical location – which is usually indicated by a (*longitude, latitude*) tuple – to a given IP address. It is of interest to many applications in which users are accessing particular services, spanning from targeted marketing to fraud prevention. With accurate geolocation of users, platforms can also provide various personalized services – for example, online targeted advertising can recommend restaurants within user's proximity. The ubiquitous location-based services have long been identified as an important challenge for the Internet.

While well-studied and impactful, IP geolocation is still a challenging problem since it is hard to collect available and reliable information. Earlier works aiming to geolocate IP addresses directly utilize the clues recorded in public databases such as WHOIS and domain name system (DNS) [26, 27]. These works maintain rule-based mapping relationships between public records and geolocations [27], which is rough and not reliable owing to the out-of-date publicly available resource. There are other research results exploiting network measurements probed by several controllable hosts such as ping value and traceroute list [17, 33]. These methods mainly treat the intersection of several circles – which are centered at the probing hosts with radii corresponding to the latency – as location boundaries (cf. Figure 1(a)).

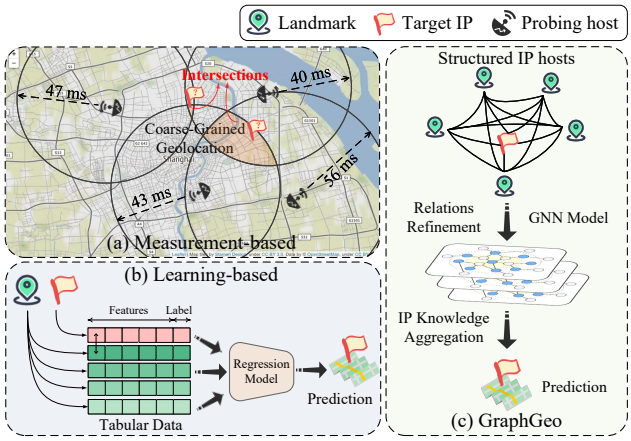


Figure 1: The core ideas of two mainstream geolocation approaches (left) and our proposed GraphGeo (right).

After decades of development, there exist various machine learning and neural network-based models for IP geolocation. Some treat measurements as features to determine a region containing the target IP as a classification task [6], or predict its longitude and latitude as a regression task [15]. Leveraging learning-based models, they can discover more beneficial clues for geolocation by integrating and analyzing multi-source data. It can easily transform features into high-dimension space for further analysis to learn common knowledge and make geolocation predictions after training with large-scale datasets.

The key ideas of the existing IP geolocation works are illustrated in Figure 1(a) and (b). Despite the encouraging results, they suffer from certain disadvantages. On the one hand, practical deployments of traditional measurement-based methods are influenced by difficult physical distance estimations and unknown spatial topological relationships. On the other hand, learning-based methods often treat IP knowledge and measurements independently as tabular data and input them into learning models such as neural networks as row vectors. However, this paradigm ignores the structured information which is strongly related to geolocations of IP hosts. In addition, the respective works do not properly account for the uncertainty of measurements caused by network congestion, jitter, topology constraints, etc., making their models not robust enough in real-world IP geolocation.

Recently, graph neural network (GNN) [21] have flourished in a series of applications. To tackle the limitations of previous works, we made a pioneering effort that exploits the graph structure to connect independent IP hosts and incorporate network topology into their features for street-level IP geolocation. There are three main challenges: **(1)** which IP hosts does a graph contain? **(2)** how to establish the appropriate graph structure? **(3)** as network measurements always hold noise and uncertainty, how to aggregate the node features robustly? To tackle these problems, we present a graph-based framework called GraphGeo that provides a full-fledged system for street-level IP geolocation. GraphGeo first employs a rule-based approach combining routers information – which is less influenced by measurement errors – to determine which graph IP hosts belong to. Subsequently, it establishes neighborhoods relationship between IP hosts from both topological and semantic perspectives to form a weighted graph and connect them. Lastly, the target IP aggregates knowledge of its neighbors with an uncertainty-aware GNN for accurate and stable outputs. Our GraphGeo solves the street-level IP geolocation problem from a novel perspective that combines knowledge of target IPs and topologies of landmarks according to the structured information, while jointly leveraging the networking rules and data-driven models. To summarize, the main contributions of this work are fourfold:

- We present GraphGeo, a novel framework which provides a complete pipeline for applying graph neural network in the street-level IP geolocation problem. It incorporates the IP host features and kinds of neighborhoods relationships, allowing us to extract the common knowledge and topology with graph structure rather than learning linear feature interactions or treating each IP host independently.
- We pinpoint the target IP address by proposing an uncertainty-aware GNN. It pays attention to the uncertainty and noise caused by network congestion and jitter that commonly exist in complicated network environments via the continuous and flexible inference in probabilistic space.
- Extensive experimental evaluations on three real-world datasets demonstrate the superiority of our framework, which not only significantly improves the accuracy of street-level IP geolocation, but also provides the interpretations of the model behaviors.
- Our framework has been successfully deployed in production on the web platform in Aiwen Tech (https://en.ipplus360.com/home/) – which provides both online API request and offline geolocation database service – for more than 6 months. To date, it has served more than 2,700 companies and clients.

2 RELATED WORK

2.1 IP Geolocation

Many researches have addressed the problem of IP geolocation prediction. Existing works can be roughly classified into two categories: data mining-based methods and measurement-based methods.

Data mining-based methods rely on mining location clues in publicly available sources to geolocate the target IP. Some of them analyze the data from IP-related databases such as WHOIS database and DNS. For example, Moore et al. [26] propose NetGeo consisting of a collection of Perl scripts for address parsing and heuristic analysis of WHOIS records to make IP geolocation predictions. Padmanabhan et al. [27] combine DNS names of target hosts and prefix information in border gateway protocol (BGP) tables to infer the geolocation. Huffaker et al. [14] collect geographic hints within the host domain and associate those strings (e.g., airport codes) with geophysical locations. Other (bodies of) works mainly extract geolocation clues from the semantic or multimedia information from the web page and social media platform. Liu et al. [25] utilize check-in records from social networks to learn a mapping model between IP, user, and geolocation. Wang et al. [34] extract location-indicating clues from web pages and locate the web servers based on the clues. Li et al. [35] and Wang et al. [23] periodically monitor websites hosting live webcams and use natural language processing

techniques to extract the IP geolocation. Despite the promising performance for large-scale IP data geolocation, existing data-mining methods are generally limited by unreliable and incomplete information due to the passive way of network data collection, which may easily lead to inaccurate geolocation results.

Measurement-based methods employ network latency and topology among probing hosts and target IPs to estimate their distance and infer the target's geolocation. This idea is based on the assumption of an existent correlation between network latency and geographical distance, which has been proven in [41]. Wang et al. [33] design a hierarchical system that begins at the large, coarse-grained scale via latency measurement and then escalates the use of external information in that scale to pinpoint the target IP. Similarly, Jiang et al. [15] first determine a big area based on delay similarity and then utilize local landmarks information via a neural network to predict the geolocation. In addition, there exists research aiming to generate more accurate latency-distance relationships. For example, Laki et al. [22] handles all the landmark points together and derives a common probabilistic delay-distance model. Hillmann et al. [12, 13] take the different Tier network levels into account and introduce a logarithmic curve to describe the delay-distance relationships. Unfortunately, latency is unstable due to the complex network environments and the topology is almost impossible to speculate. These issues need to be properly considered to enable high-quality street-level IP geolocation prediction.

2.2 Graph Neural Networks

Using graph neural networks (GNN) to analyze structured data when an underlying graph describes the dependencies between items has become one of the most popular data mining topics. Early explorations were mainly based on spectral graph theory [4] to aggregate knowledge from adjacent nodes/edges for rich relational information learning. Among these works, graph convolutional network (GCN) [21], graph attention network (GAT) [31], and variants [11, 20, 37] have gained a lot of attention due to their performance improvements in various domains. GNNs have been widely applied in many application domains, such as recommendation systems and traffic flow [7, 9, 16, 32].

As for the geolocation area, GNNs have been used to combine both text and network context to locate the user in social networks [28, 39]. Although GNN methods have a strong ability to handle structured data and are intuitively suitable for IP geolocation prediction, to our knowledge, there is no previous work exploiting GNNs for the IP geolocation task. The main obstacle is the unknown structured information and uncertain network measurements, making it challenging to capture topological interactions among nodes when using GNNs – which usually require deterministic relations for feature aggregation.

What separates our work from the existing approaches is that we seamlessly integrate graph neural networks into the street-level IP geolocation task. We establish appropriate relationships between IP hosts based on their knowledge and network topology as edges in the graph. Moreover, we incorporate uncertainty – a commonly encountered phenomenon in complicated network environments – and infer representation in probabilistic space for accurate and robust IP geolocation.

3 DATA AND PROBLEM DEFINITION

In this section, we first describe our data and the corresponding collection methods, and then proceed with formally defining the street-level geolocation problem.

3.1 Data Collection

Our proprietary ground truth dataset contains millions of IP addresses with their own knowledge (e.g., autonomous systems (AS) and WHOIS data), network measurement (e.g., ping value and traceroute list), and geolocations marked by longitude and latitude.

The knowledge of IP hosts is obtained by lookup in publicly available databases such as WHOIS. We collect the network measurements by executing probing commands, such as *"ping"* and *"tracert"*, on multiple probing hosts located in different regions (at night to minimize the impact of network congestion). The IP geolocations are derived with two main approaches: (1) We collect geolocations from devices with global positioning sensors (GPS) uploaded by users with permission when they check in the online platform. It contains both mobile and fixed broadband IP addresses since users often connect their mobile phones to WiFi; (2) We select automated or semi-automated crowdsourcing markets to place a job where users can actively submit their IP addresses and accurate locations. After the proof of audit workload, we collect the results into our real-world dataset. We employ the IP addresses whose geolocations have no significant change in historical records (detailed filtering is described in Appendix A) to conduct experiments.

3.2 Problem Definition

We now define our street-level IP geolocation problem, which aims at inferring the geolocation of target IP address with a set of measurements and available IP hosts knowledge. Given N landmarks $L = \{L_1, L_2, \cdots, L_N\}$ with their knowledge $\mathbf{X}_L^k$, network measurements $\mathbf{M}_L$ and locations $\mathbf{Y}_L$ marked by longitude and latitude, we are interested in pinpointing a target IP with its own knowledge $\mathbf{X}_T$ and network measurements $\mathbf{M}_T$ via a data-driven model, which can be formulated as:

$$\widehat{\mathbf{Y}}_T = \text{GraphGeo}(\mathbf{X}_L, \mathbf{M}_L, \mathbf{Y}_L, \mathbf{X}_T, \mathbf{M}_T; \Theta)\,, \tag{1}$$

where $\widehat{\mathbf{Y}}_T = (\widehat{lon}_T, \widehat{lat}_T)$ is a tuple denoting the predictive longitude and latitude of target IP, and Θ denotes all hyper-parameters and parameters of the proposed model GraphGeo.

4 METHODOLOGY

We now present the details of GraphGeo.

4.1 Overview

For the street-level IP geolocation problem, we make the first attempt that establishes graphs with structured data containing spatial topology of the network, and geolocate target IP hosts via continuously refining their knowledge. Specifically, we propose a graph-based framework GraphGeo whose overall framework is shown in Figure 2. We can observe that GraphGeo consists of three main modules: (1) IP hosts clustering with topology; (2) IP hosts connecting; (3) IP knowledge aggregation with uncertainty. The first one, aiming at locating the target IP in a graph containing a series of landmarks, is the premise of subsequent processing. The

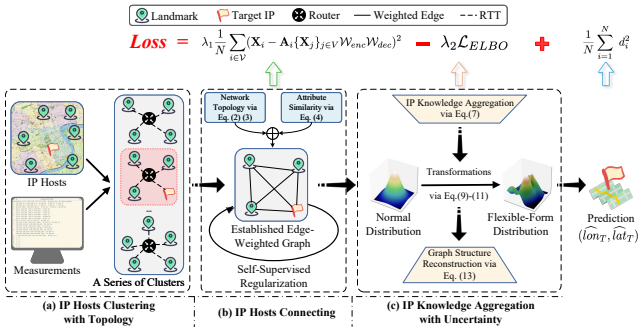


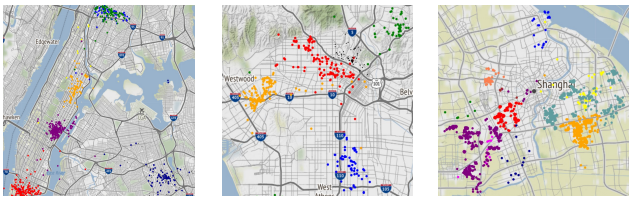
Figure 2: Overview of the proposed GraphGeo framework.

second one establishes topological and senior semantic relations to connect the target IP and landmarks and form a weighted graph. The last module pinpoints the target IP via features aggregation with an uncertainty-aware graph learning layer. We will describe the details of each module in the following section.

4.2 IP Hosts Clustering with Topology

Determining which graph the target IP belongs to is the first overarching challenge, which requires meeting two conditions: (1) there exist direct or indirect topological correlations between every two IP hosts; (2) IP hosts in a same graph should be close in the real world. The former provides the structured information between the nodes, which supports the feasibility of further graph analysis. The existing literature demonstrates the superiority of narrowing the scope of geolocation step-by-step in our problem [15, 33]. Thus, the latter collecting target IP and its near landmarks is beneficial for street-level geolocation.

The majority of prior works achieve clustering with ping response latency between landmarks and target IP, using topological approach [33] or unsupervised methods such as k-nearest neighbours [15]. However, long-distance measurements such as latencies are heavily influenced by varying network conditions, which easily leads to inaccuracy with a large error bound.



(a) New York (b) Los Angeles (c) Shanghai

Figure 3: Distributions of IP hosts colored by the common last-hop router in their traceroute list.

In this paper, we design rule-based clustering methods, which cluster IP hosts with common last hop in their traceroute lists into a graph efficiently and stably. Specifically, we use the *"tracert"* tool on probing hosts located in different cities to lookup the last-hop router. If there exist multiple last-hop routers, we select the router with the most negligible latency. If firewall strategies hide the last-hop router, we will take the last visible one. The algorithm of the clustering is described in Appendix B. On the one hand, as we can observe in Figure 3, the IP hosts with the common last-hop router usually have close physical distance from each other due to the regional management strategy on the Internet. On the other hand, we are able to build exact topological relationships between the target IP and landmarks since we can take the common router as the bridge to connect them. This provides a rough area containing target IP and the feasibility of analyzing the precise geolocation with both the IP host's knowledge and the topological relationships via a graphic structure.

4.3 IP Hosts Connecting

Existing literature that utilizes the interactions between IP hosts usually pinpoints the target IP by taking the intersection of multiple circles as shown in Figure 1(a) [12]. However, there will be kinds of possibilities for network topologies leading it unworkable and they cannot make a better prediction with useful IP hosts knowledge. In GraphGeo, we are aiming to establish an effective graphic structure to connect independent IP hosts for street-level IP geolocation.

Denote the undirected weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, where $\mathcal{V}$ represents a set of nodes (corresponding to IP hosts), $\mathcal{E}$ represents the edges whose weights are described by the (entries of the) matrix $\mathbf{A}$, and N is the number of nodes. We establish our graph for geolocation with two assumptions: (1) in the graph structure, the larger edge weight reflects the stronger correlations between two nodes; (2) IP hosts with stronger correlations have closer distances between their geolocation coordinates. In a word, we shall let edges weights between geographically close IP hosts become larger. We define our edges from two perspectives, i.e., *network topology* and *attribute similarity*, to achieve this purpose.

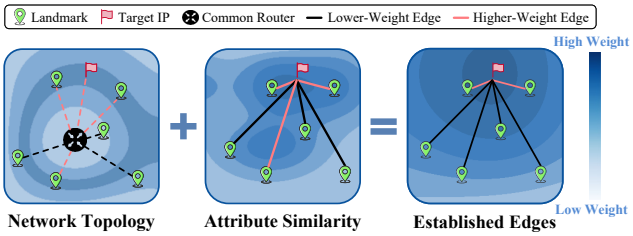


Figure 4: The edge establishment with two perspectives, i.e., network topology and attribute similarity.

4.3.1 Network Topology. We first utilize the existing topology from network measurements between IP hosts and their common router to establish edges. As shown in Figure 4, IP hosts in the same graph surround a common router. We determine the landmarks with similar radius to the target IP since some of them have close geographical distance to it. Therefore, we naturally construct edge weights that are inversely proportional to the distance difference between the target IP and landmarks. It can be described as:

$$\mathbf{A}^{nt}_{t,l} = \exp(-\Delta^2(t,l)) = \exp(-|d(t,r) - d(l,r)|^2), \qquad (2)$$

where t, l, and r separately denote a target IP, a landmark, and the common router in a graph. ϵ is a hyperparameter to control the

range. $\Delta(t, l)$ calculates the difference of radial distance between t and l via $d(\cdot, r)$ – which maps the network measurements into geographical distance.

As known, the correlation between round-trip time (RTT) and geographic distance approximately follows a logarithmic curve [12, 13]. Thus, we implement the mapping function as follows:

$$d(i, r) = \alpha\left(\beta \log RTT(i, r) + Hop(i, r)\right) + \gamma, \tag{3}$$

where i denotes any landmarks or the target IP, $RTT(i, r)$ and $Hop(i, r)$ represent latency and routing hops between the IP host and common router. α, β, and γ are parameters obtained by a fully connected network with the corresponding network measurement $\mathbf{M}_L$ and $\mathbf{M}_T$ input. With Eq.(2) and (3), we are able to calculate edge weights about network topology $\mathbf{A}^{nt}$ between every two IP hosts.

4.3.2 Attribute Similarity. Geographical topologies of IP hosts in a small area are very hard to speculate. Thus, we additionally introduce the second horizon to better connect IP hosts and describe their relationships. Besides, allocation of IP address can be conjectured with its knowledge and feature interactions. A simple case is that IPs are usually allocated by their virtual address order and the relationship between the addresses of two IP hosts can reflect their topology to some extent. Thus, we further exploit the attribute similarities between the target IP and landmarks as another form of edge weights.

Given a knowledge of the target IP t and a landmark l, we calculate their similarity by carrying dot-product on their feature vectors:

$$\mathbf{A}^{as}_{t,l} = \frac{\exp(\{\mathbf{X}_t, \mathbf{X}_l\}\mathcal{W}_q \cdot (\{\mathbf{X}_t, \mathbf{X}_l\}\mathcal{W}_k)^{\mathrm{T}})}{\sum_{l' \in L} \exp(\{\mathbf{X}_t, \mathbf{X}_{l'}\}\mathcal{W}_q \cdot (\{\mathbf{X}_t, \mathbf{X}_{l'}\}\mathcal{W}_k)^{\mathrm{T}})} \tag{4}$$

where $\cdot^{\mathrm{T}}$ represents transposition, $\{\mathbf{X}_t, \mathbf{X}_l\} \in \mathbb{R}^{2d_{\mathrm{X}}}$ consists of the IP host knowledge of target IP and landmark, $\mathcal{W} \in \mathbb{R}^{d_{\mathrm{X}} \times d_m}$ denotes learned parameters, and d_m is the metric dimensions. This formula takes original IP features into metric space and extracts the similarity interactively for better semantic correlations.

4.3.3 Self-Supervised Weights Fusion. With two edge weights from different perspectives, we incorporate both the topological relationships and IP hosts knowledge within structured information. However, the current structure we establish is a complete graph. This will not only consume luxurious computing resources but also take some unrelated nodes into consideration, which leads to the inefficiency and inaccuracy. Thus, we introduce a mechanism to control the sparsity of the final graph:

$$\mathbf{A}_{t,l} = \begin{cases} \mathbf{A}^{nt}_{t,l}/\kappa + \mathbf{A}^{as}_{t,l}, & \text{if } \mathbf{A}^{nt}_{t,l}/\kappa + \mathbf{A}^{as}_{t,l} > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where κ and ϵ are parameters and thresholds to control the sparsity. As sizes of graph are different, we employ the ϵ-th quantile edge weights of each node as thresholds.

It is difficult for the model to converge while learning both graph structure and geolocation prediction simultaneously. Besides, a graph structure that is suitable for predicting the node features is also suitable for predicting the node labels [8]. Therefore, we improve the quality of learned graph and convergence by adding an additional self-supervised regularization term:

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{N} \sum_{i \in \mathcal{V}} (\mathbf{X}_i - \mathbf{A}_i\{\mathbf{X}_j\}_{j \in V}\mathcal{W}_{enc}\mathcal{W}_{dec})^2, \tag{6}$$

where $\mathcal{W}_{enc}$ and $\mathcal{W}_{dec}$ denote the learned parameters of encoder and decoder network. Minimizing this term motivates the graph to reconstruct nodes feature and further benefits our street-level IP geolocation task.

As for edge weights between landmarks, we employ the same methods used in calculating relationships between the target IP and landmarks. For edge weights between target IPs, we set them as 0 which will be consistent with the real application of IP geolocation.

4.4 IP Knowledge Aggregation with Uncertainty

Theoretically, we can put all landmarks and target IP nodes into a GNN to output the final geolocation. However, such a direct modeling ignores the widespread uncertainty in measurements caused by complicated network conditions. Therefore, we present an uncertainty-aware graph neural network to break through this limitation for more robust and accurate geolocation. About edges, we employ the weights obtained in the previous section, and about nodes, we use the knowledge and longitude and latitude of each IP, among which we use mask $(0, 0)$ instead of the geolocations of target IP nodes. We first aggregate IP host knowledge and extract the deterministic representation containing both nodes and structured information via a 2-layer GCN:

$$\mathrm{GCN}(\mathcal{X}, \mathbf{A}) = \widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}} \cdot \mathrm{Relu}(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathcal{X}\mathcal{W}_1) \cdot \mathcal{W}_2, \tag{7}$$

where $\mathcal{X} = \{\mathbf{X}, \mathbf{Y}'\}$ is the concatenation of IP hosts knowledge and their geolocation, $\mathbf{Y}'$ means we mask the geolocation of target IP, and $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\widehat{\mathbf{D}}$ is the degree matrix of $\widehat{\mathbf{A}}$. After that, all IP nodes in this graph have aggregated the knowledge from their neighbors weighted by $\mathbf{A}$. However, all representations cannot escape the instability owing to network environments. Thus, we introduce stochastic noise and build a probabilistic distribution to reduce the negative impacts from these factors and keep geolocation results accurate and robust. For this, we first establish a simple posterior distribution to describe the uncertainty of the graph structure via reparameterization trick [19]:

$$q(\mathbf{Z}|\mathcal{X}, \mathbf{A}) = \prod_{i=1}^{N} \mathcal{N}(z_i|\mu_i, \sigma_i^2), \tag{8}$$

where $\mu = \mathrm{GCN}_\mu(\mathcal{X}, \mathbf{A})$, $\sigma = \log(1 + exp(\mathrm{GCN}_\sigma(\mathcal{X}, \mathbf{A})))$ are produced by two GCNs GCN_μ and GCN_σ with different parameters. $\mathbf{Z}$ is the stochastic variable which follows Gaussian distribution and denotes the latent states of all IP nodes with uncertainty. z_i, μ_i, and σ_i are the i-th vector of themselves respectively.

Gaussian distribution models are suitable for the majority of cases since their data usually distributes evenly around the average. However, in our problem, we aim to describe the uncertainty of network measurements beyond the Gaussian distribution assumption. For this characteristic, we continue our inference to make our current posterior distribution $q(\mathbf{Z}|\mathcal{X}, \mathbf{A})$ approximate the actual situations with a more flexible form. We introduce a continuous-form normalizing flow [3] for the probabilistic distribution transformation. Compared with traditional normalizing

flows [13], it calculates transformations efficiently in a continuous way with linear time complexity, which makes our method more applicable for timely IP geolocation prediction. Let $\mathbf{Z}(t_0) = \mathbf{Z}$ and $q(\mathbf{Z}(t_0)|\mathcal{X}, \mathbf{A}) = q(\mathbf{Z}|\mathcal{X}, \mathbf{A})$ be the initial random variable and probabilistic distribution, the transformations can be present as:

$$\frac{d\mathbf{Z}(t)}{dt} dt = f(\mathbf{Z}(t), t; \theta_f), \tag{9}$$

$$\mathbf{Z}(t_1) = \mathbf{Z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{Z}(t), t; \theta_f) dt, \tag{10}$$

$$\log q(\mathbf{Z}(t_1)|\mathcal{X}, \mathbf{A}) = \log q(\mathbf{Z}(t_0)|\mathcal{X}, \mathbf{A}) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{Z}(t)}\right) dt, \tag{11}$$

where t_1 is a hyperparameter representing the span of transformations, $f(\cdot)$ is a fully connected layer with learned parameters θ_f which provides the derivative of $\mathbf{Z}$ with respect to t, Eq. (11) is the instantaneous change of variables formula [3], and $\mathrm{Tr}(\cdot)$ is the trace function. Now we can optimize the final posterior distribution by maximizing the evidence lower bound (ELBO) as follows:

$$\begin{aligned}\mathcal{L}_{ELBO} =& \mathbb{E}_q \left[\log p(\mathbf{A_{i,j}}|\mathbf{Z}_i(t_1), \mathbf{Z}_j(t_1)) + \log p(\mathbf{Z}(t_1))\right] \\ &+ \mathbb{E}_q \left[\int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{Z}(t)}\right) dt - \log q(\mathbf{Z}(t_0)|\mathcal{X}, \mathbf{A})\right].\end{aligned} \tag{12}$$

The term $\log p(\mathbf{A}_{i,j}|\mathbf{Z}_i(t_1), \mathbf{Z}_j(t_1))$ represents the reconstruction loss between established graph structure and the reconstructed one from distribution, which is a significant term to improve the quality of posterior distribution. We employ an efficient calculation inspired by [24]:

$$\begin{aligned}&\log p(\mathbf{A}_{i,j}|\mathbf{Z}_i(t_1), \mathbf{Z}_j(t_1)) \\ &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{A}_{i,j} \log(1 + \exp(H(f(\mathbf{Z}_i(t_1)), \mathbf{Z}_j(t_1); \theta_p))),\end{aligned} \tag{13}$$

where $H(\cdot)$ is a Heaviside step function and $f(\cdot)$ is a fully connected network to reconstruct the graph from latent states. Besides, we follow the general approaches described in [13, 20] for other terms' calculations. After that, we obtain a more accurate posterior probabilistic distribution and the latent variable $\mathbf{Z}(t_1)$ includes the uncertain information.

4.5 Prediction & Training

We now can output the longitude and latitude of target IP immediately with the combination of $\mathrm{GCN}(\mathcal{X}, \mathbf{A})$ and $\mathbf{Z}$ via a MLP:

$$(\widehat{lon}_T, \widehat{lat}_T) = \mathrm{MLP}(\mathrm{GCN}(\mathcal{X}, \mathbf{A}), \mathbf{Z}; \theta_m). \tag{14}$$

We train our framework with stochastic gradient descent (SGD) using Adam optimizer. Our objective is composed of three parts: (1) minimizing the loss between predicted geolocation and the ground truth; (2) minimizing the self-supervised regularization term in Eq.(6) for a better graph structure; and (3) maximizing the ELBO loss in Eq.(12) for an accurate posterior distribution. The training loss of our GraphGeo can be present as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} d_i^2 + \lambda_1 \mathcal{L}_{\mathcal{G}} - \lambda_2 \mathcal{L}_{ELBO}, \tag{15}$$

where N_t is the number of target IPs, $\lambda \geq 0$ is a tradeoff parameter, and d_i denotes the great circle distance between predicted geolocation $(\widehat{lon}_T, \widehat{lat}_T)$ and the ground truth (lon_T, lat_T). The algorithm and complexity analysis of the optimization are described in Appendix C.

5 EXPERIMENTS

We now present the extensive experiments that we conducted to validate the behavior of the proposed GraphGeo. We first make comparisons with the state-of-the-art baselines and ablation study analysis to validate the performance of our model. Subsequently, we explore the parameter sensitivity and feature importance to provide more interpretability. We note that example use-case is discussed in Appendix D.

5.1 Experimental Settings

Datasets. We use three large-scale real-world street-level IP geolocation datasets collected from three metropolises, i.e., New York, Los Angeles, and Shanghai, which consists of 91808, 92804, and 126,258 IP addresses, respectively. For the data processing, we randomly select 70% IPs for training and the remaining 30% for testing. In the training process, we take 70% IP as landmarks and 30% as target IPs. In the test process, we treat the training set as landmarks and others as target IPs to report the results.

Baselines. We compared our GraphGeo with the following state-of-the-art baselines in three categories:

- **CBG** [10]: establishes a continuous space and infers the geographic locations of Internet hosts using multilateration positioning with distance constraints.
- **TBG** [17]: is a topology-based geolocation approach that converts topology and communication delay into a set of constraints to geolocate routers and Internet hosts simultaneously.
- **TNN** [15]: is a two-layer neural network that first determines a rough region and then narrows the scope of the target IP for street-level IP geolocation.
- **TLP** [5]: proposes the concept of IP range interpolation and combines it with traceroute latency neighbors for IP geolocation.
- **SVR**: is the regression type of support vector machine (SVM) that has been used to improve IP geolocation performance [38].
- **LightGBM** [18]: is an advanced gradient boosting decision tree method and has achieved promising results on many algorithm competitions.
- **AutoInt** [1]: is an automatic feature interaction learning model for tabular data. It exploits a self-attentive neural network [30] for feature engineering and missing value prediction.
- **GAE** [20]: learns latent variables and makes variational inference with the probabilistic form to handle graph-structured data.
- **GAT** [31]: employs an attention mechanism to exploit the relationships between nodes when aggregating the features, which can be directly applied to capture the feature interactions.
- **GraphSAGE** [11]: uses graph edges to sample the neighboring nodes, and then aggregates features by a multi-layer structure.
- **RGCN** [40]: adopts Gaussian distributions as the hidden representations in each convolutional layer to improve the robustness.
- **CompGCN** [29]: is a graph model that leverages a variety of composition operations from knowledge graph embedding techniques to embeds both nodes and relations in a relational graph.

Table 1: Performance comparisons on street-level IP geolocations. Best performance is in bold font and the second best results are underlined. All results are measured in kilometers (km).

Type	Method	New York			Los Angeles			Shanghai		
		RMSE	MAE	Median	RMSE	MAE	Median	RMSE	MAE	Median
IP Geolocation Methods	CBG	19.73	15.22	12.40	21.39	17.86	14.03	38.20	30.54	26.57
	TBG	16.00	13.15	10.62	17.78	15.53	12.29	31.62	26.31	22.87
	TNN	4.262	3.169	2.511	7.403	3.962	2.568	11.163	8.136	6.094
	TLP	4.025	3.674	2.503	6.973	3.818	2.441	9.172	7.320	5.429
Tabular Learning Methods	SVR	11.20	9.867	8.721	14.33	12.59	11.49	23.52	20.40	18.28
	AutoInt	5.197	3.852	3.044	8.134	4.401	2.941	12.59	8.743	6.485
	LightGBM	4.826	3.754	2.961	7.027	4.135	2.687	11.50	8.562	6.349
Graph Methods	GAE	4.237	3.204	2.582	7.143	3.913	2.545	11.01	8.296	6.179
	GAT	3.981	2.863	2.104	6.751	3.790	2.367	9.823	7.882	5.706
	GraphSAGE	3.762	2.710	2.013	6.512	3.691	2.307	9.412	7.765	5.503
	RGCN	3.093	2.455	1.692	6.210	3.461	2.621	9.211	7.428	5.405
	CompGCN	3.012	2.443	1.530	6.207	3.455	2.471	9.163	7.864	5.129
	Graphormer	2.964	2.315	1.418	6.224	3.483	2.470	9.125	6.917	4.510
Ours	**GraphGeo**	**2.201**	**1.329**	**0.890**	**6.032**	**3.318**	**1.587**	**7.800**	**5.184**	**3.259**

- **Graphormer** [36]: builds upon the standard Transformer architecture, and could attain excellent results on a broad range of graph learning tasks.

Parameter Settings. In the *IP hosts connecting* module, we set the metric dimensions $d_m = 64$, i.e., $\mathcal{W}_q, \mathcal{W}_k \in \mathbb{R}^{2d_X \times 16}$ in Eq.(4). Parameters that control the graph sparsity (cf. Eq.(5)) ϵ are defined as 60%. Besides, we set the dimensions as 32 for all latent states in *IP knowledge aggregation with uncertainty* module. In Eq.(10)–Eq.(12), we employ integral terms for posterior distribution inference. We use fourth-order Runge-Kutta (RK-4) [2] for its calculation and apply the adjoint method [3] for backpropagation with less memory-resources consumption. For the MLP in prediction (cf. Eq.(14)), we exploit a 2-layer structure whose width of the inner layer is 32. Tradeoff parameters λ_1 and λ_2 (cf. Eq.(15)) are separately defined as 0.8 and 0.5 to balance the scale of each regularization term. We train our framework with learning rate searched within $[10^{-6}, 10^{-3}]$ for different datasets and halve it every 50 epochs. The training will halt when parameter updates no longer yield improves on performance for 100 epochs.

Evaluation Protocols. We evaluate the performance of our model GraphGeo using three widely used metrics for IP geolocation and regression tasks: median, mean absolute error (MAE), and root mean square error (RMSE). We also draw the cumulative distribution function curve (CDF) with respect to the predictive errors to show the quality of geolocation results.

5.2 Performance Comparison

Table 1 shows experimental results of different models across three IP geolocation datasets. We can see that our GraphGeo achieves the best performance in terms of all metrics. *CBG* and *TBG* only rely on latency measurements to make predictions, which are limited by unstable network conditions and inaccurate latency-distance mapping and, as a result, lead to substantial geolocation errors. In other words, they are not suitable for street-level geolocation.

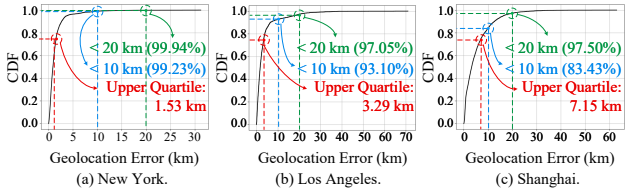


Figure 5: The CDF curves w.r.t. geolocation errors of GraphGeo across three datasets.

SVR, *LightGBM*, and *AutoInt* are all learning-based methods being capable of handling tabular data. These methods capture the feature interactions directly from the data, and their geolocation performance also heavily relies on the accuracy of measurements. Thus, they can only determine a rough area containing the target IP but cannot accurately pinpoint the IPs. This result implies that precise IP geolocation requires the rule-based method to continuously infer geographical locations, especially with uncertain and inaccurate Internet measurements. It also explains the motivation of this work to exploit the network rules and the relations between IP hosts for street-level IP geolocation.

TNN is a two-tiered model that first localizes a target IP within a rough area and then fine-tunes the results by exploiting the landmarks. Similarly, a series of graph models use the measurements as edge weights for feature aggregations, which can fully make use of the information of landmarks while accounting for their interactive relationships. Thus, these GNN-based models attain better results than other baselines. Nevertheless, they also ignore the networking rules that play essential roles in street-level IP geolocation.

In contrast, our model provides an applicable geolocation framework that simultaneously explores IP knowledge and considers the uncertainty of the networking information, enabling fine-grained geolocations even with considerable noise in the data. Besides, our model exploits GNN for street-level IP geolocation that pays extra attention to both the spatial network topology and semantic

similarities between IP hosts. Figure 5 depicts the CDF curves of GraphGeo on three datasets, which indicates that our method can localize more than half of the IP addresses within 5km errors across all datasets. For example, more than 60% of geolocations results are within 1km distance of the ground truths. And, more than 80% of IPs are pinpointed within 10km in Shanghai, while the percentages are greater than 90% in New York and Los Angeles.

5.3 Ablation Study

We conduct the ablation study to validate the effectiveness of the key modules of GraphGeo. We name GraphGeo without a specific module as follows: (1) **GraphGeo-N** removes network topology perspective when establishing edges; (2) **GraphGeo-A** removes attribute similarity perspective when establishing edges; (3) **GraphGeo-S** removes self-supervised regularization term when optimizing the graph; (4) **GraphGeo-U** removes uncertainty-aware probabilistic inference in IP knowledge aggregation.

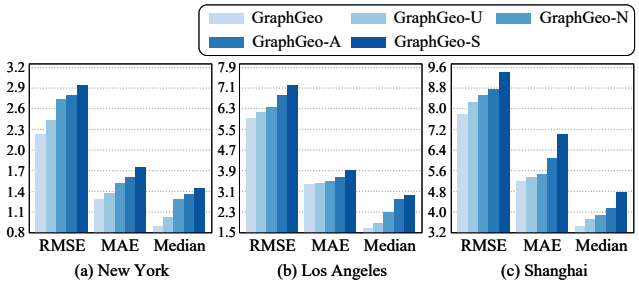


Figure 6: Ablation studies across three datasets.

As shown in Figure 6, the four main modules indeed contribute to the final geolocation. The edge weights produced from two perspectives and the self-supervised regularization term play significant roles, which supports our intuition to find out the landmarks relevant to the target IP that can accurately calculate the weighted geolocations. Besides, uncertainty-aware inference also improves the performance since it handles the noisy data with probability.

5.4 Feature Importance

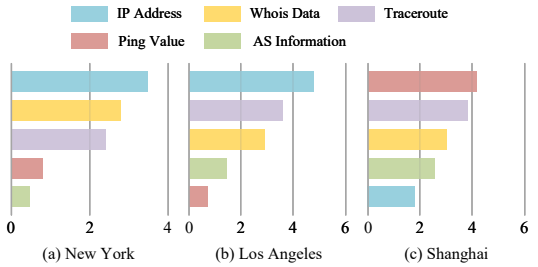


Figure 7: Effect of each class of features on geolocation. We record the logarithmic relative changes on RMSE metric.

To investigate the importance of different features and validate the interpretability of GraphGeo , we compare the effect of each class of features on the geolocation results. The various features of each IP address that we use can be grouped into five classes: *IP address*, *AS information*, *WHOIS data*, *ping values*, and *traceroute measurements*. With the optimized model GraphGeo, we mask each class of features with zero vectors and obtain the importance value by observing their relative performance changes. The results are shown in Figure 7. We can observe that there are both commonalities and differences in the effect of features on three datasets. For example, traceroute information significantly impacts IP geolocations across three datasets, which supports the importance of (using) this data. We can also see less influence of AS information since it is hard to extract geolocation-related clues. Interestingly, ping value plays an important role on the Shanghai dataset, while it has little influence on the other two. The main reason is that the IP hosts clustering categories in Shanghai usually cover greater areas. Therefore, the ping values in a greater area usually contain larger differences and thus are more critical for the final geolocation outputs. These results demonstrate that GraphGeo has the capability to find effective features and makes use of them for accurate street-level geolocations.

5.5 Parameter Sensitivity

We further explore the influence of the following important parameters in GraphGeo.

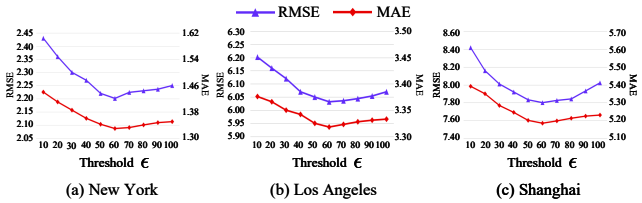


Figure 8: Influence of the parameter sparsity threshold.

Sparsity threshold ϵ. ϵ is the threshold that controls the graph sparsity (cf. Eq.(5)). Neighbors (i.e., landmarks) with edge weights greater than ϵ are kept while others are discarded. We tune ϵ within range [10, 100] with 10 intervals and results are shown in Figure 8. We can observe that the performance of GraphGeo improves a lot first and then declines with ϵ growing and achieves the best at $\epsilon = 60$. The reason is fewer neighbors lack enough knowledge to determine the geolocation of the target IP, while tremendous neighbors lead to the homogenization of edge weights and geolocation prediction close to the center of landmarks. It also explains why the negative influence on MAE is slighter than that on RMSE. Besides, performance with very few landmarks drops but our model still performs better than other baselines in most metrics. It proves the superiority in the few shot situation, where we will continue paying attention in our future work.

Loss tradeoff λ. We employ two parameters λ_1 and λ_2 to keep all loss terms in the same scale (cf. Eq.(15)). We jointly tune two parameters and plot the results shown in Figure 9. We can easily determine appropriate tradeoff parameters, i.e., $\lambda_1 = 0.8$ and $\lambda_2 = 0.5$. Besides, λ_1 plays an important role in the geolocation results, which demonstrates that our self-supervised regularization is beneficial for establishing a good graph structure. Similarly, we can observe from the changes of λ_2 that inference with uncertainty

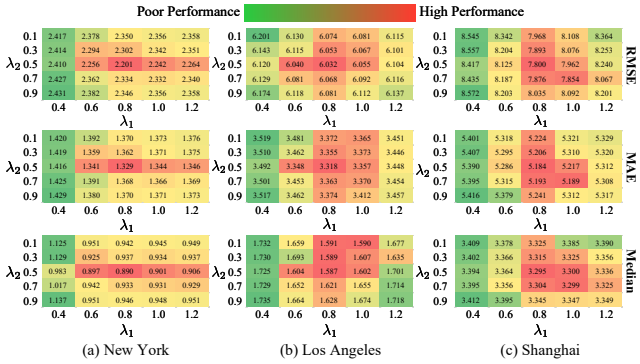


Figure 9: Influence of the parameter loss tradeoff.

does improve the performance since it has the ability to handle the widely existing noisy and unstable network measurements.

6 CONCLUSION

We presented GraphGeo, a novel framework for street-level IP geolocation, incorporating IP hosts knowledge and their neighborhoods relationships, with an uncertainty-aware graph neural network. It provides a complete processing stack on applying network and spatial topology to connect IP hosts for high-quality geolocation. Extensive experiments on real-world datasets validate the effectiveness of GraphGeo, which shows the superiority to existing measurement-based and learning-based approaches on geolocating large-scale IP addresses. Moreover, our framework has been deployed in Aiwen Tech and provided kinds of services for more than 6 months.

ACKNOWLEDGEMENTS

This work was supported in part by National Natural Science Foundation of China (Grant No.62176043 and No.62072077), Sichuan Science and Technology Program (No.2022YFSY0006), National Science Foundation SWIFT (Grant No.2030249), and Hong Kong RGC TRS T41-603/20-R.

REFERENCES

[1] Sercan Ö. Arik and Tomas Pfister. 2021. TabNet: Attentive Interpretable Tabular Learning. In *AAAI*. 6679–6687.
[2] Uri M Ascher, Steven J Ruuth, and Raymond J Spiteri. 1997. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.* (1997), 151–167.
[3] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *NeurIPS*. 6572–6583.
[4] Fan RK Chung and Fan Chung Graham. 1997. *Spectral graph theory*. American Mathematical Soc.
[5] Ovidiu Dan, Vaibhav Parikh, and Brian D Davison. 2021. IP Geolocation Using Traceroute Location Propagation and IP Range Location Interpolation. In *WWW*. 332–338.
[6] Brian Eriksson, Paul Barford, Joel Sommers, and Robert Nowak. 2010. A learning-based approach for IP geolocation. In *PAM*. 171–180.
[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*. 417–426.
[8] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks. In *NeurIPS*.
[9] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. 2021. Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. arXiv:2109.12843
[10] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. 2006. IEEE/ACM Trans. Netw. *Transactions On Networking* (2006), 1219–1232.
[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[12] Peter Hillmann, Lars Stiemert, Gabi Dreo, and Oliver Rose. 2020. On the Path to High Precise IP Geolocation: A Self-Optimizing Model. arXiv:2004.01531
[13] Peter Hillmann, Lars Stiemert, Gabi Dreo Rodosek, and Oliver Rose. 2015. Modelling of IP Geolocation by use of Latency Measurements. In *CNSM*. 173–177.
[14] Bradley Huffaker, Marina Fomenkov, and KC Claffy. 2014. DRoP: DNS-based router positioning. *Comput. Commun. Rev.* (2014), 5–13.
[15] Hao Jiang, Yaoqing Liu, and Jeanna N Matthews. 2016. IP geolocation estimation using neural networks with stable landmarks. In *INFOCOM WKSHPS*. 170–175.
[16] Weiwei Jiang and Jiayun Luo. 2021. Graph Neural Network for Traffic Forecasting: A Survey. arXiv:2101.11174
[17] Ethan Katz-Bassett, John P John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. 2006. Towards IP geolocation using delay and topology measurements. In *SIGCOMM*. 71–84.
[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*. 3146–3154.
[19] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.
[20] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. arXiv:1611.07308
[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[22] Sándor Laki, Péter Mátray, Péter Hága, Tamás Sebők, István Csabai, and Gábor Vattay. 2011. Spotter: A model based active geolocation service. In *INFOCOM*. 3173–3181.
[23] Qiang Li, Zhihao Wang, Dawei Tan, Jinke Song, Haining Wang, Limin Sun, and Jiqiang Liu. 2021. GeoCAM: An IP-Based Geolocation Service Through Fine-Grained and Stable Webcam Landmarks. *IEEE/ACM Trans. Networking* (2021), 1798–1812.
[24] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.
[25] Hao Liu, Yaoxue Zhang, Yuezhi Zhou, Di Zhang, Xiaoming Fu, and KK Ramakrishnan. 2014. Mining checkins from location-sharing services for client-independent ip geolocation. In *INFOCOM*. 619–627.
[26] David Moore, Ram Periakaruppan, Jim Donohoe, and Kimberly Claffy. 2000. Where in the world is netgeo.caida.org?. In *INET*.
[27] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. 2001. An investigation of geographic mapping techniques for internet hosts. In *SIGCOMM*. 173–185.
[28] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Semi-supervised User Geolocation via Graph Convolutional Networks. In *ACL*. 2009–2019.
[29] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.
[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.
[31] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.
[33] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. 2011. Towards Street-Level Client-Independent IP Geolocation.. In *NSDI*. USENIX.
[34] Yucheng Wang, Hongsong Zhu, Jinfa Wang, Jie Liu, Yong Wang, and Limin Sun. 2020. XLBoost-Geo: An IP Geolocation System Based on Extreme Landmark Boosting. arXiv:2010.13396
[35] Zhihao Wang, Qiang Li, Jinke Song, Haining Wang, and Limin Sun. 2020. Towards IP-based geolocation via fine-grained and stable webcam landmarks. In *WWW*. 1422–1432.
[36] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation?. In *NeurIPS*.
[37] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*. 793–803.
[38] Qian Zhao, Fei Wang, Can Huang, and Chuan Yu. 2020. Improving IP geolocation databases based on multi-method classification. In *ASID*. 44–48.
[39] Ting Zhong, Tianliang Wang, Fan Zhou, Goce Trajcevski, Kunpeng Zhang, and Yi Yang. 2020. Interpreting Twitter user geolocation. In *ACL*. 853–859.
[40] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *SIGKDD*. 1399–1407.
[41] Artur Ziviani, Serge Fdida, José F De Rezende, and Otto Carlos MB Duarte. 2005. Improving the accuracy of measurement-based geographic location of Internet hosts. *Comput. Networks* (2005), 503–523.

Algorithm B1 Rule-based IP hosts clustering.

Input:
Traceroute lists of a series of IP hosts from probing hosts.
Output:
IP clustering categories containing the hosts.

Initialize the IP clustering category.
foreach *IP host* **do**
  Initialize an empty last router list, latency list and hop list.
  foreach *router list* **do**
    Append last visible router into last route list.
    Append corresponding network latency and routing hops into latency list and hop list, respectively.
  end for
  Find the last router with the lowest latency.
  Take the IP host into category marked by the last router and record corresponding routing information.
end for

A DATA FILTERING

We collect the geolocation of each IP address via kinds of approaches, e.g., uploaded GPS information and crowdsourcing markets. However, multiple historical geolocation records of an IP address may be different owing to various reasons. For example, a user using cellular network is assigned an IP address by a base station. The geolocation records of this IP will change within the base station coverage. We select some IP addresses and illustrate their historical geolocation records in Figure A1, where we can observe geolocations of some IPs (e.g., light blue points) change a lot while some (e.g., red and orange points) are stable. Therefore, we select IPs with more than 4 historical records which changes within 1 kilometer to form the dataset for our street-level geolocation problem.

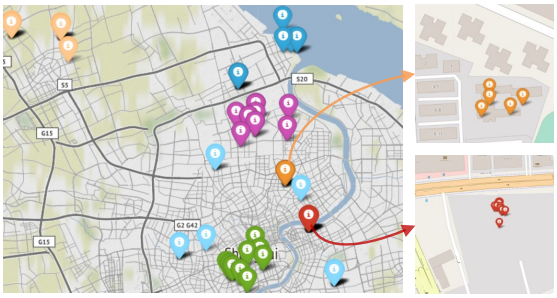


Figure A1: Distribution of historical geolocations colored by different IP addresses.

B IP HOSTS CLUSTERING

We design rule-based methods to cluster IP hosts efficiently and stably, which avoids the negative influence of quality fluctuations in networks. Figure B1 illustrates the data and its structure required for IP hosts clustering. For each IP host, there exist multiple traceroute lists obtained by probing hosts located in different regions. Each router list holds a sequence consisting of passing routers and corresponding latency. Given these data, we provide the Algorithm B1 to describe the process of IP hosts clustering.

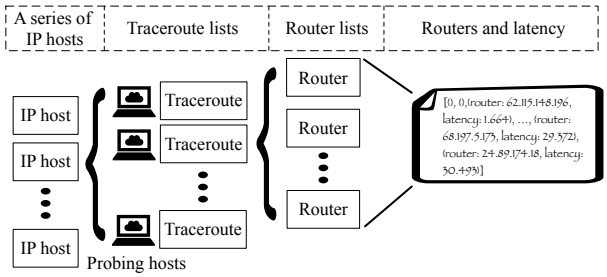


Figure B1: Data structure of traceroute lists of IP hosts.

C TRAINING & COMPLEXITY ANALYSIS

In this section, we provide the description of the complete training procedure and corresponding complexity analysis.

C.1 Training Algorithm

We summarize the pseudo-code of the training process of proposed GraphGeo in Algorithm C1.

Algorithm C1 Training of GraphGeo.

Input:
Knowledge and network measurements of the landmarks and target IP $\mathbf{X}_L$, $\mathbf{M}_L$ and $\mathbf{X}_T$, $\mathbf{M}_T$;
Geolocations of landmarks $\mathbf{Y}_L$;
Sparsity threshold, loss tradeoff and other hyperparameters.
Output:
Geolocation predictions of target IP $\widehat{\mathbf{Y}}_T = (\widehat{lon}_T, \widehat{lat}_T)$;
Optimized parameters Θ.

for *a batch* $\in$ *training set* **do**
  Determine which graph the target IP belongs to via rule-based IP clustering (cf. Algorithm 1).
  Obtain network topology and attribute similarity weights between IP hosts and the target IP via Eq.(2) and Eq.(4).
  Fuse the edge weights from two perspectives via Eq.(5).
  Calculate the graph structure loss $\mathcal{L}_{\mathcal{G}}$ as a self-supervised regularization term via Eq.(6).
  Aggregate IP hosts knowledge with a GCN via Eq.(7).
  Establish a flexible posterior distribution $q(\mathbf{Z}|\mathcal{X}, \mathbf{A})$ with continuous transformations via Eq.(8)-(11).
  Calculate ELBO loss as a regularization term via Eq.(12).
  Predict geolocation of the target IP $\widehat{\mathbf{Y}}_T$ via Eq.(14).
  Add MSE loss to $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{ELBO}$ via Eq.(15).
  Back-propagate the gradients and update the learned parameters Θ with Adam optimizer.
end for

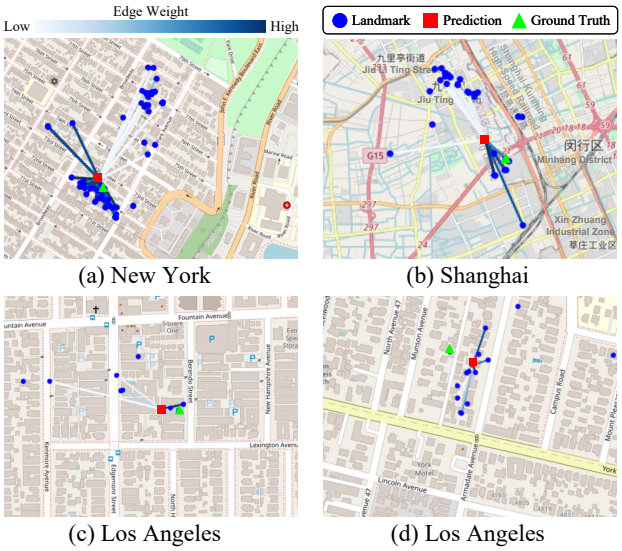
Table C1: Complexity analysis of GraphGeo and its modules.

Module	Time Complexity
IP hosts clustering	$O(NL_r)$
IP hosts connecting	$O(N^2 d_{\mathbf{X}} d_m)$
IP knowledge aggregation	$O(Nd_z^2)$
GraphGeo	$O(NL_r + N^2 d_{\mathbf{X}} d_m + Nd_z^2)$

C.2 Complexity Analysis

We analyze the time complexity of the main modules of the proposed model GraphGeo. The *IP hosts clustering* module based on predefined rules does not require training. Its loops w.r.t. IP hosts can run in parallel and the complexity becomes $O(NL_r)$, where L_r denotes the length of the router lists in traceroute. The *IP hosts connecting with topology* module establishes graph structure from two perspectives, i.e., network topology and attribute similarity. Network topology taking fully connected layers outputs numerical value as parameters incurs $O(N^2 d_{\mathbf{X}})$ time complexity, where $d_{\mathbf{X}}$ is the dimension of IP host knowledge. The time complexity of attribute similarity calculating edge weights is linear with $O(N^2 d_{\mathbf{X}} d_m)$. The *IP knowledge aggregation with uncertainty* module includes two steps: (1) aggregating IP host knowledge and building a normal distribution posterior consumes $O(Nd_z^2)$ time complexity; (2) a series of transformations with continuous-form normalizing flow (cf. Eq.(11)) also require $O(Nd_z^2)$, where d_z denotes the dimension of latent states $\mathbf{Z}$. Time complexities of proposed GraphGeo and its each module are summarized in Table C1.

D CASE STUDY



(a) New York (b) Shanghai

(c) Los Angeles (d) Los Angeles

Figure D1: Case study.

We now illustrate and analyze the use of GraphGeo on street-level IP geolocation. As an example of the use of GraphGeo in geolocation predictions, we show the geolocation prediction, the ground truth, and established relations between target IP and landmarks in Figure D1. In most cases, there are appropriate relationships between the target IP and landmarks, which follows our assumption on edge weights. Landmarks with close distance usually holds high edge weights, and the target IP is surrounded by landmarks. After IP knowledge aggregation with weighted structure, our graph-based framework GraphGeo can perform well on street-level geolocation tasks. However, we can also observe that the predictive errors will increase when the target IP is located at the corner of clustering category (cf. Figure D1(d)). This may be because these kinds of target IPs lack sufficient neighbors to restore spatial topologies, which leads to inaccuracy. Taking some measures such as specific rules or training penalties may be effective, and we are currently investigating such aspects.