TrustGeo: Uncertainty-Aware Dynamic Graph Learning for Trustworthy IP Geolocation

Wenxin Tai
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Bin Chen
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Fan Zhou*
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Ting Zhong
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Goce Trajcevski
Iowa State University
Ames, Iowa, USA

Yong Wang
Hong Kong University of Science and Technology
HongKong, China

Kai Chen
Hong Kong University of Science and Technology
HongKong, China

ABSTRACT

The rising popularity of online social network services has attracted a lot of research focusing on mining various user patterns. Among them, accurate IP geolocation is essential for a plethora of location-aware applications. However, despite extensive research efforts and significant advances, the "accurate and reliable" desideratum is yet to be achieved at a higher quality level. This work presents a graph neural network (GNN)-based model, called TrustGeo, for trustworthy street-level IP geolocation. A distinct and important aspect of TrustGeo is the incorporation of sources of uncertainty in the learning process. The results of our extensive experimental evaluations on three real-world datasets demonstrate the superiority of our framework in significantly improving the accuracy and trustworthiness of street-level IP geolocation. Our code and datasets are available at https://github.com/ICDM-UESTC/TrustGeo.

CCS CONCEPTS

• **Information systems** → **Location based services**; • **Networks** → **Location based services**.

KEYWORDS

IP geolocation, trustworthy system, graph neural networks, network uncertainty, evidential deep learning

ACM Reference Format:
Wenxin Tai, Bin Chen, Fan Zhou, Ting Zhong, Goce Trajcevski, Yong Wang, and Kai Chen. 2023. TrustGeo: Uncertainty-Aware Dynamic Graph Learning for Trustworthy IP Geolocation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599920

*Corresponding Author.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599920

1 INTRODUCTION

Accurately determining the geographic location of Internet hosts is an essential functionality for many location-aware applications such as spectrum allocation, smart cities, Geoweb, etc. [11, 12, 31]. The existing popular client-dependent geolocation techniques are typically relying on location data obtained via sources such as global positioning systems (GPS), cell towers, and Wi-Fi [29]. However, these techniques often rely on clients' voluntary participation (i.e., the server obtains geo-location data only if the users explicitly report it), which, however, may yield privacy concerns [13]. Hence, they are inapplicable when location-based access is restricted – aside from the fact that accurate geolocation is still a challenge for a large number of devices (desktops and laptops) that use wired access (i.e., without GPS, cellular, or Wi-Fi).

A large body of client-independent approaches has been proposed to improve geolocation accuracy. Some works [25, 26] use diverse information mined from the Internet and pinpoint IPs by analyzing clues from the massive collected data; others [10, 31] locate addresses based on active network delay measurements, using the multilateration technique with some predefined delay-distance conversion functions. Recently, neural networks (NNs) have been used to prevent empirical errors caused by expert-designed hypothetical rules. Some works [14, 36] automatically learn the mapping from the attribute information of the target host to the geographic location, while others [7, 34] establish the connections between the target host and surrounding coordinates by mining the neighborhood information using graph neural networks (GNNs) [35].

One of our previous works, GraphGeo [34], is a state-of-the-art method for IP geolocation. It establishes neighborhood relationships between IP hosts from topological and semantic perspectives. Then the knowledge of its neighbors is aggregated via attentive

graph learning. This approach has been shown to reduce geolocation errors significantly [34]. However, we have identified *two primary challenges* that need to be addressed during the deployment of our geolocation system: (1) The effectiveness of GraphGeo depends heavily on the quality of the constructed IP graph. If the graph structure is unreliable (noisy node attributes or inaccurate edge weights), or sparse (few adjacent landmarks), then the graph learning algorithm may learn biased knowledge, reducing the trustworthiness of the geolocation system [28]. (2) As a service provider, we hope to find an efficient way to update the algorithm progressively. While GraphGeo only provides an estimated location of the target IP, it does not offer developers any feedback to improve the system.

To overcome these challenges, we propose TrustGeo, which incorporates uncertainty-aware dynamic graph learning to enhance both the accuracy and reliability of the geolocation system. TrustGeo also generates a confidence score (a.k.a. epistemic uncertainty) that provides developers with extra signals to monitor the system's performance. The core of TrustGeo is to incorporate sources of uncertainty into the learning process and treat the graph-based IP geolocation as a multi-view decision fusion task: We first define the knowledge used for geolocation as being obtained from two aspects – the information from the target IP itself and the collaborative information from its neighbors. Then we calculate the uncertainty for each view and use these uncertainties as criteria to adaptively fuse information from different views. This approach ensures that the geolocation system remains effective even under challenging scenarios, and helps developers make informed decisions when updating the system.

We further make some improvements to the construction of the IP graph. Specifically, GraphGeo constructs the graph using only landmarks and target hosts. It directly uses the difference between the delays from the landmark and the target host to the router as an approximation of the delay between the landmark and the target IP host. However, in most cases, such delay calculation is inaccurate, resulting in sub-optimal prediction performance [5]. In this work, we propose Star GNN, which can incorporate the feature information of IP hosts and neighbor relationships in a more natural way. A star graph consists of a *star* node and multiple *satellite* nodes, where the star node has connections with all satellite nodes and can pass messages to them. Since a common router can communicate with the target IP and its surrounding landmarks, we view the common router as the star node and treat the target IP and landmarks as satellite nodes.

We conduct extensive experiments on three large-scale, real-world IP datasets. The results demonstrate that TrustGeo significantly improves the accuracy and reliability of street-level IP geolocation and is more efficient than state-of-the-art baselines. We would like to highlight some specific benefits that customers can benefit from our proposed trustworthy IP geolocation system. (1) Our system incorporates uncertainty learning and graph neural networks to enhance the accuracy of the IP geolocation, resulting in more precise location estimates compared to the previous version of our system (GraphGeo). This can have significant implications for businesses that rely on location-based services, such as targeted advertising, e-commerce, and logistics. (2) Our system is designed to be more robust to "various noises", which is a key challenge in IP geolocation. By addressing this challenge, our proposed method offers an increased level of trustworthiness, which can be especially important for applications related to security, fraud detection, and law enforcement.

2 RELATED WORK

We now review the two main categories of related literature and position our work in that context.

(1) IP Geolocation. There are three main IP geolocation paradigms in the literature.

– *Web mining approaches* use diverse information mined from the Internet, including web page content [33], WHOIS databases [25], reverse DNS [26], postal addresses [11], users' registration records [21, 26], and social graphs [32]. For example, *NetGeo* [25] parses the IP address and analyzes WHOIS records to predict IP locations. *Checkin-Geo* [21] leverages the location data that users are willing to share in location-sharing services and logs of user logins from PCs for geolocation. *GeoCAM* [33] periodically monitors websites hosting live webcams to extract location information. Web mining-based approaches are generally limited by unreliable and incomplete information due to the passive network data collection that may easily lead to inaccurate geolocation [34].

– *Active Network Delay Measurements* is at the majority of IP geolocation research works rely on. The rationale behind these approaches is the positive correlation between network delay and geographic distance [31, 37]. The earlier work *GeoPing* [26] sends ICMP packets from geographically distributed probing hosts to the target IP, which is localized by the location of the closest landmark server. *CBG* [10] creates circles on the surface of the earth around each landmark server, and uses multilateration to infer the location of the target IP. Complementary to network delay, *TBG* [16] combines network delay with topology measurements using traceroute from landmark servers to the target IP for location estimation. *Spotter* [19] is a probabilistic density model that derives a common delay-distance model and utilizes the highest probability density to localize the target hosts. This line of methods obey several delay-distance hypothetical rules. However, they relied on the experience of networking experts, thus limiting the generalization capabilities.

– *Machine Learning & Deep Learning* have recently been exploited by a growing number of researchers for locating IP addresses. *NN-Geo* [14] collects RTTs (round trip times) from multiple observers as features, and uses RBF (radial basis function) neural network to pinpoint IPs. *GNN-Geo* [7] reformulates IP geolocation as an attributed graph node regression problem and designs a GNN-based framework to exploit surrounding knowledge from the target IP host. *GraphGeo* [34] establishes neighborhood relationships between IP hosts from both topological and semantic perspectives to form a weighted graph, and aggregates the knowledge of its neighbors with an uncertainty-aware GNN for IP geolocation.

(2) Uncertainty in Deep Learning. Deep learning tools have gained tremendous attention in applied machine learning, but the ones for regression and classification do not capture model uncertainty. Instead of solely optimizing larger models for increased performance, uncertainty deep learning [1] focuses on how these models can be equipped with the ability to estimate their own confidence. According to [6], there are two axes of NN uncertainty

that can be modeled: (1) uncertainty in the data, called aleatoric uncertainty, and (2) uncertainty in the prediction, called epistemic uncertainty. Aleatoric uncertainty captures noise inherent in the observations. This could be for example network uncertainty and fluctuations resulting in uncertainty that cannot be reduced even if more data were to be collected. Epistemic uncertainty accounts for uncertainty in the model parameters – which can be used to describe the confidence of what our model learns. While representations of aleatoric uncertainty can be learned directly from data [2], there exist several approaches for estimating epistemic uncertainty, such as Bayesian NNs, evidential learning, etc. Bayesian models [30] offer a mathematically grounded framework to reason about model uncertainty but usually come with a prohibitive computational cost. Some methods for uncertainty estimation leverage the concept of Bayesian learning, e.g., *Monte Carlo Dropout* [9] or *Ensembling* [20]. These techniques require training a large number of model instances or fail to produce diverse predictions. Recently, an alternative class of models based on the concept of *Evidential Deep Learning* [2, 23] has emerged, which allows uncertainty estimation in a single model and forward pass by parameterizing distributions over distributions.

Summary of Differences: Our work differs from existing studies in two aspects.

– *Router-level graph learning.* TrustGeo is a GNN-based method that effectively narrows the region by clustering the IPs at the router level. There exist approaches that also constrain region via the last-hop router. For example, *NCRGeo* [4] which calculates the distance between the router and landmarks, and then uses multilateration to obtain geolocation of the common router while directly taking the location of the router as the estimated location of the target IPs. *XLBoost-Geo* [32] extracts location indicating clues from web pages and locates the web servers through recurrent neural networks. Both *NCRGeo* and *XLBoost-Geo* fail to exploit neighborhood relationships. The most recent work *GraphGeo* [34] constructs a graph to exploit topological and semantic information, whereas only the target host and landmarks are considered. Our novel star GNN explicitly takes the common router into account, which prevents inaccurate relative distance estimation.

– *Uncertainty estimation.* TrustGeo includes a novel uncertainty learning algorithm that takes the network measurement uncertainty and dynamic variations of the IP graph into account, making it more trustworthy in real-world IP geolocation services. The importance of uncertainty learning for IP geolocation has rarely been discussed in previous literature. Note that *Spotter* and *GraphGeo* (respectively using probabilistic density function and normalizing flow to model the uncertainty) only model the aleatoric uncertainty from data but neglect the epistemic uncertainty from the model, which affects the reliability and accuracy of geolocation.

3 METHODOLOGY

We now formally define the problem and present the details of TrustGeo.

3.1 Problem Definition

Considering the universality of the algorithm, we follow previous settings [7, 10, 16, 34] and define geolocation task under the passive pattern (i.e., a set of passive landmarks with known locations), to which measurements can be made, but cannot originate any measurements of their own. In this setting, we have a set of reference hosts with known locations (i.e., landmarks). Some landmarks are active and can issue probes (make network measurements to each other and to the passive landmarks and targets), while some cannot (i.e., passive). More specifically, when we locate a target, we assume that the locations of the remaining targets are known, thus we can use the remaining targets as passive landmarks[1].

Definition 3.1 (Active/Passive Landmarks)**.** To bootstrap the system, there need reference hosts with known locations that refer to landmarks [16]. Among them, active landmarks are landmarks (reference nodes) that can make network measurements to each other (passive landmarks/targets), while passive landmarks cannot issue measurements, but can be probed by active landmarks.

We now define the street-level IP geolocation problem, aiming to infer the geolocation of the target IP addresses with a set of measurements and available IP hosts knowledge.

Definition 3.2 (Network Measurements)**.** There are mainly two categories of measurement – the delay measurements and Internet route measurements [16]. Specifically, network measurements are collected by executing probing commands, such as *"ping"* and *"traceroute"*, on multiple probing hosts (active landmarks) located in different regions.

Definition 3.3 (Attribute Knowledge)**.** Information collected or retrieved from WHOIS[2] will be used as the attribute knowledge [34]. In this work, attribute knowledge includes 14 dimensions: IPv4 address, ID of the autonomous system where IP is located country (resp. state/province/city) of IP location, etc.

Definition 3.4 (IP Geolocation)**.** Given N landmarks $\{\mathbf{l}_i\}_{i=1}^N$ with their knowledge $\{\mathbf{x}_i\}_{i=1}^N$, network measurements $\{\mathbf{m}_i\}_{i=1}^N$ and locations $\{\mathbf{y}_i\}_{i=1}^N$ marked by longitude and latitude, we seek to pinpoint a target IP with its own knowledge $\mathbf{x}_T$ and the network measurements $\mathbf{m}_T$ via a data-driven model:

$$\widehat{\mathbf{y}}_T = \text{TrustGeo}(\{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{m}_i\}_{i=1}^N, \{\mathbf{y}_i\}_{i=1}^N, \mathbf{x}_T, \mathbf{m}_T; \boldsymbol{\theta}), \quad (1)$$

where $\widehat{\mathbf{y}}_T=(\widehat{lon}_T, \widehat{lat}_T)$ denotes the predictive geographical locations of target IP, and $\boldsymbol{\theta}$ denotes all the learnable parameters.

It is important to note that the information on WHOIS might be inaccurate or outdated sometimes[3]. Besides, when the network experiences congestion or fluctuations, measurement data such as ping and traceroute may also be inaccurate. However, we allow the existence of such inaccuracy as it reflects the quality of the collected data and further highlights the significance of uncertainty learning in this study. In other words, the inaccuracy of attribute knowledge and measurement data can serve as inherent real-world noises, and aleatoric uncertainty is introduced to estimate it (see Section 3.3 for details).

[1] This widely used and overconfident assumption is not always valid in practice. We challenge this assumption and focus on either quantifying or compensating for network uncertainty (cf Sec. 3.4).
[2] https://www.whois.com
[3] https://www.icann.org/resources/pages/inaccuracy-2013-03-22-en

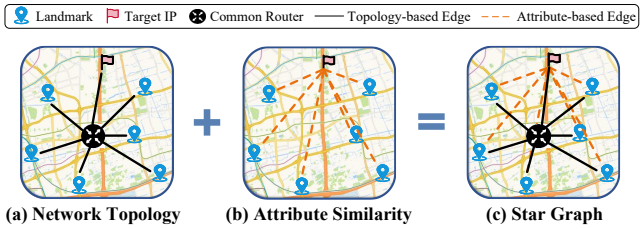


Figure 1: Illustration of building the star graph. We view the common router as a star node and treat the target IP and landmarks as satellite nodes. The proposed star graph takes a "multi-pronged" approach in which known attributes are considered, and at the same time looks for structure in the IP network that can be exploited.

3.2 Geolocation via Graph Neural Network

The majority of prior works realize clustering relying on *ping* response latency between landmarks and target IP, using either topological approach [16] or unsupervised methods (e.g., k-nearest neighbors [8]). However, long-distance measurements such as communication latency are influenced by varying network conditions, yielding inaccurate estimates with large error bound [5, 14, 31].

Following [4, 32, 34] in connecting IP hosts with common last hop, we use the *"traceroute"* tool to probe hosts located in different cities and look up the last-hop router(s). If there exist multiple last-hop routers, we select the router with the smallest (i.e., most negligible) latency, for twofold reasons: (1) 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 [31]; (2) We can build the topological relationships between the target IP and landmarks since we can take the common router as the bridge to connect them. In practice, if the firewall hides the last-hop router, we consider the last visible one.

A recent work [34] constructs the graph with landmarks and target hosts. The difference of the delays from the landmark and the target host to the router is treated (roughly) as the delay between the landmark and the target host. However, in most cases, such delay calculation is inaccurate, resulting in sub-optimal prediction performance [5]. To address this, we introduce the star graph to avoid the negative impact of insufficient routing knowledge when establishing the graph. Specifically, a star graph consists of a *star* node and *satellite* nodes, where the star node has connections with all satellite nodes and can pass messages to them. Since a common router can communicate with the target IP and its surrounding landmarks, we view the common router as the star node. The target IP and landmarks will be treated as satellite nodes. Figure 1 illustrates the procedure of building the star graph.

Star connections. Considering we have the location of landmarks and round-trip time (RTT) between hosts (landmarks, target host) and the common router, we can directly construct topology connections – we naturally construct edge weights that are inversely proportional to the delay between the common router and hosts:

$$\boldsymbol{A}_{R,*} = \exp(-\gamma \cdot (\alpha \cdot \mathrm{RTT}(*, R) + \beta)), \tag{2}$$

where $*$ denotes an arbitrary satellite node, R denotes the star node (router), and α, β, γ are trainable parameters.

Satellite connections. Upon obtaining the graph that consists of a series of landmarks and the target IP, with the common router as the bridge to connect them, suppose that the geolocation of the router is $\widehat{\mathbf{y}}_R$. According to the topology between the router and landmarks, we rely on the constraint of topology measurement – i.e., the geographic location of the target IP $\widehat{\mathbf{y}}_T$ is somewhere on a circle with a radius of $d(T, R)$ centered on the routing (star) location $\widehat{\mathbf{y}}_R$. That is, the geolocation of the target IP cannot be accurately located with measurements only in the last-hop-based router-centric graph. As noted in [34], geographically close nodes usually have similar properties. This observation enables analyzing the precise geolocation with the host's attribute knowledge and topological relationships via the graph structure. Hence, we can establish direct connections between different satellite nodes as we have attribute knowledge of the target IP and landmarks. Given the knowledge of the target IP T and a landmark l, we calculate their similarity via the dot-product of their feature vectors:

$$\boldsymbol{A}_{T,l} = \exp(\boldsymbol{v}^{\mathrm{T}} \sigma(\boldsymbol{W}_1\{\mathbf{x}_T, \mathbf{m}_T\} + \boldsymbol{W}_2\{\mathbf{x}_l, \mathbf{m}_l\} + \boldsymbol{b})), \tag{3}$$

where $\{,\}$ is the concatenation operation, σ denotes the *sigmoid* function, $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{(d_x+d_m)\times(d_x+d_m)}$, and $\boldsymbol{b}$, $\boldsymbol{v} \in \mathbb{R}^{(d_x+d_m)}$ are trainable matrices and vector, respectively. As a result, Eq. (3) connects the target host with its surrounding landmarks via the semantic similarity $\boldsymbol{A}_{T,l}$.

Learning nodes representations. For an established star graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each node $v \in \mathcal{V}$ will be paired with a node representation $\mathbf{h}_v$ initialized as $\mathbf{h}_v^0 = \{\mathbf{x}_v, \mathbf{m}_v\} \in \mathbb{R}^{d_x+d_m}$. However, the attribute information of the star node is missing. To initialize the star node, we apply average pooling on representations of satellite nodes. During the i-th iteration, each $\mathbf{h}_v^i$ is updated using v's neighborhood information as:

$$\mathbf{h}_v^{(i)} = \mathrm{Update}^{(i)}\left(\mathbf{h}_v^{(i-1)}, \mathrm{Agg}^{(i)}\left(\mathbf{h}_u^{(i-1)}, \mathbf{A}_{u,v}\right)\right), \tag{4}$$

where $u \in \mathcal{N}_v$, Agg() is a trainable function that maps the set of node representations to an aggregated vector, Update() is another trainable function that maps both v's current representation and the aggregated vector to update v's representation. After 2 iterations of Eq. (4), the updated node representation of the target IP host is $\boldsymbol{z} = \mathrm{Concat}(\{\boldsymbol{h}_v^i\}_{i=0}^2)$, where node v is the target IP host. As a result, we learn the representation of target IP through three perspectives: (1) zero-order self-information, (2) first-order aggregated information from neighbors via attribute similarity, and (3) second-order aggregated information from neighbors via measurement information (using the common router as the transition node).

3.3 Network Uncertainty Measurement

So far, we have updated the representation of the target IP node through graph learning. Normally, we can consider the IP geolocation task as a deterministic regression problem, which commonly optimizes the sum of the square errors between the estimation $\widehat{\mathbf{y}}_T$ and the ground truth $\mathbf{y}_T$ – we can predict the geographic location of the target IP immediately via multi-layer perceptron (MLP), using mean squared error (MSE) as the cost function. However, this

approach does not explicitly model any underlying noise or uncertainty when making its estimation. In real-world scenarios, the information on WHOIS may sometimes be inaccurate or outdated. Additionally, when the network experiences congestion or fluctuations, measurement data such as ping and traceroute may not be reliable. These two factors weaken the quality of the constructed IP graph (noisy node representation and inaccurate edge weights), leading to inaccurate geolocation results. It would be beneficial for the algorithm to detect such circumstances and alert users that the model is uncertain about its current prediction result.

To this end, we consider the geolocation problem where the observed locations, $\mathbf{y}_T = (lon_T, lat_T)$, are drawn i.i.d from a Gaussian distribution, with unknown mean and variance $(\{\mu_o, \mu_a\}, \sigma^2)$, which we seek to probabilisticaly estimate. We model this by placing a prior distribution on $(\{\mu_o, \mu_a\}, \sigma^2)$. For simplicity, we use μ to denote $\{\mu_o, \mu_a\}$, where μ_o denotes the mean estimation of the lontitude, and μ_a denotes the mean estimation of the latitude. As suggested in [2], we place a Gaussian prior[4] on unknown means μ and an Inverse-Gamma prior on the unknown variance σ^2. Suppose we have target sample $\mathbf{y}_T$ need to be estimated, we have:

$$\mathbf{y}_T \sim \mathcal{N}(\mu, \sigma^2) \tag{5}$$

$$\mu \sim \mathcal{N}(\gamma, \sigma^2 v^{-1}) \qquad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta), \tag{6}$$

where $\Gamma(\cdot)$ is a Gamma function, and $\gamma \in \mathbb{R}^2, v > 0, \alpha > 1, \beta > 0$. Our goal is to estimate a posterior distribution $q(\mu, \sigma^2|\mathbf{y}_T)$. According to [2], to obtain an approximation for the posterior, the estimated distribution can be factorized such that $q(\mu, \sigma^2) = q(\mu)q(\sigma^2)$. Thus, our approximation takes the form of the Gaussian conjugate prior, the Normal Inverse-Gamma (NIG) distribution:

$$p(\{\mu, \sigma^2\} \mid \Omega) = \frac{\beta^\alpha \sqrt{v}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta + v(\gamma-\mu)^2}{2\sigma^2}\right\}, \tag{7}$$

where $\Omega = \{\gamma, v, \alpha, \beta\}$. The aleatoric uncertainty, also known as statistical or data uncertainty, captures noise inherent in the observations and is representative of unknowns towards experimental environments. The epistemic (or model) uncertainty, describes the estimated confidence of the model prediction. Given a NIG distribution, we can compute the prediction, aleatoric, and epistemic uncertainty as:

$$\underbrace{\mathbb{E}[\mu] = \gamma}_{\text{prediction}}, \quad \underbrace{\mathbb{E}\left[\sigma^2\right] = \frac{\beta}{\alpha - 1}}_{\text{aleatoric}}, \quad \underbrace{\text{Var}[\mu] = \frac{\beta}{v(\alpha-1)}}_{\text{epistemic}}. \tag{8}$$

Having established the use of an evidential distribution to account for both aleatoric and epistemic uncertainty, we next outline the method for training a model to produce the hyperparameters of this distribution. To make it clear, we have divided the training process into two separate stages: (1) increasing model evidence to support our observations, and (2) reducing evidence (increasing uncertainty) when predictions are incorrect. In general terms, we can view (1) as a way of adapting our data to the evidential model while (2) enforces a prior to eliminate inaccurate evidence and increase uncertainty.

(1) Maximizing the model fit. According to Bayesian probability theory, the "model evidence", or marginal likelihood, is defined as the likelihood of an observation $\mathbf{y}_T$, given the evidential distribution parameters Ω and is computed by marginalizing over the likelihood parameters (μ, σ^2):

$$p\left(\mathbf{y}_T \mid \Omega\right) = \int_{\sigma^2=0}^{\infty} \int_{\mu=-\infty}^{\infty} p\left(\mathbf{y}_T \mid \mu, \sigma^2\right) p\left(\mu, \sigma^2 \mid \Omega\right) \mathrm{d}\mu \mathrm{d}\sigma^2. \tag{9}$$

Evaluating model evidence can be challenging as it requires integrating the dependence on latent model parameters. However, when a NIG evidential prior is applied to the Gaussian likelihood function, there does exist an analytical solution [2]:

$$p\left(\mathbf{y}_T \mid \Omega\right) = \text{St}\left(\mathbf{y}_T; \gamma, \frac{\beta(1+v)}{v\alpha}, 2\alpha\right), \tag{10}$$

where $\text{St}\left(\mathbf{y}_T; \mu_{\text{St}}, \sigma^2_{\text{St}}, v_{St}\right)$ is a Student-t distribution evaluated at $\mathbf{y}_T$ with location μ_{St}, scale σ^2_{St}, and v_{St} degrees of freedom. We use the negative logarithm of model evidence as the training objective $\mathcal{L}^{\text{NLL}}$:

$$\mathcal{L}^{\text{NLL}}(\boldsymbol{\theta}) = \frac{1}{2}\log\left(\frac{\pi}{v}\right) - \alpha\log(\omega) + \log\left(\frac{\Gamma(\alpha)}{\Gamma\left(\alpha + \frac{1}{2}\right)}\right) + \left(\alpha + \frac{1}{2}\right)\log\left((\mathbf{y}_T - \gamma)^2 v + \omega\right), \tag{11}$$

where $\omega = 2\beta(1+v)$. This loss guides NN to output parameters of a NIG distribution to fit the observations by maximizing the model evidence.

(2) Minimizing evidence on errors. We now explain how to regularize training by applying an incorrect evidence penalty (i.e., high uncertainty prior) to reduce evidence on incorrect predictions. It is important to note that regularization terms should reduce the weight of evidence where the geolocation prediction is inaccurate while having little or no impact on evidence prediction where the prediction is close to the ground truth. To achieve this, we formulate an evidence regularizer [2], $\mathcal{L}^{\text{R}}$, scaled on the error of the prediction:

$$\mathcal{L}^{\text{R}}(\boldsymbol{\theta}) = |\mathbf{y}_T - \mathbb{E}\left[\mu\right]| \cdot \phi = |\mathbf{y}_T - \gamma| \cdot (2v + \alpha). \tag{12}$$

This loss imposes a penalty whenever there is an error in the prediction and scales with the total evidence of the inferred posterior [2]. Conversely, even if the amount of predicted evidence is large, it will not be penalized (or little influenced) as long as the prediction is close to the ground truth.

Training. The total loss, $\mathcal{L}(\boldsymbol{\theta})$, consists of the two loss terms for maximizing and regularizing evidence, scaled by a regularization coefficient λ, is defined as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}^{\text{NLL}}(\boldsymbol{\theta}) + \lambda\mathcal{L}^{\text{R}}(\boldsymbol{\theta}), \tag{13}$$

where λ trades off uncertainty inflation with model fitting. Setting $\lambda = 0$ yields an over-confident estimate while setting λ too high results in over-inflation (cf. Sec 4.5). In practice, NN with parameters $\boldsymbol{\theta}$ is trained to output Ω for each target sample. We use the mean

[4] As per Bayesian Inference, it is common to model a regression problem using a normal likelihood for uncertainty estimation, yielding > 1 potential choice for a prior distribution. Prior studies have used the Normal-Inverse Gamma distribution, induced a scaled inverse-$\mathcal{X}^2$ posterior or a Normal-Wishart prior. Our own experiments suggest that a normal prior for the mean and an inverse Gamma prior for the variance can provide effective uncertainty estimation.

value γ to pinpoint the target IP and use variance value $\frac{\beta}{v(\alpha-1)}$ to illustrate the model's uncertainty. By taking uncertainty estimation into consideration, our method can make corresponding responses to network variations and outdated knowledge sources. Detailed experiments can be found in Sec 4.1.

3.4 Quantifying Dynamic Graph Uncertainty

In addition to the network variations and outdated knowledge sources, limited neighbors will also affect the performance of graph learning – the model might learn biased knowledge from the sparse graph. In fact, the graph sparsity problem occurs frequently in practice. For instance, when the system experiences a cold start or is used in rural areas, there may be few neighbor landmarks available related to the target IP. This can result in poor stability for GNN-based geolocation algorithms. To our knowledge, most previous IP geolocation methods [7, 10, 16, 34] assume the availability of landmarks with known locations. While it could be more reasonable to use few hosts, these methods consider that *ALL* hosts in their dataset (except the target one) act as landmarks to facilitate the GNN-based geolocation approaches. In this work, we challenge this assumption and propose a dynamic graph uncertainty awareness method to improve the accuracy and stability of the geolocation algorithm when the quality of the constructed IP graph is unstable.

As the graph learning algorithm in Sec. 3.2 has two views (self-information view $\mathbf{h}_v^0$ and graph learning view $\{\boldsymbol{h}_v^k\}_{k=1}^2$)), traditional graph learning algorithms usually integrate the original data or preprocessed features by simple deterministic fusion – which ignores the view-specific uncertainty and is unaware of the quality variation of different views for different samples. To this end, we develop a novel information-fusion algorithm that can capture the view-specific uncertainty and use these uncertainties as criteria to adaptively fuse information from different views. Specifically, we separately apply evidential uncertainty learning (Sec 3.3) for the self-information view and graph learning view. Accordingly, we can obtain two NIG distributions: $NIG(\gamma_1, v_1, \alpha_1, \beta_1)$, and $NIG(\gamma_2, v_2, \alpha_2, \beta_2)$. After that, we need to study how to fuse these distributions into a new distribution. Particularly, the fusion strategy needs to obey the following rules: (1) The new distribution should also be a NIG distribution – so that we can estimate the overall uncertainty of our geolocation system. (2) The fusion strategy should take uncertainty into consideration, that is, the higher the uncertainty of the view, the lower the coefficient of the view when fused. (3) The fusion strategy should satisfy the *Commutativity* – so that we can guarantee that the fusion strategy has no correlation with the order of views. Inspired by recent work in multi-modal learning [22], we introduce the NIG summation operation proposed in [27] to approximately solve this problem. Given two NIG distributions from different views, the definition of the summation of these two NIG distributions is:

$$NIG(\gamma, v, \alpha, \beta) := NIG(\gamma_1, v_1, \alpha_1, \beta_1) \oplus NIG(\gamma_2, v_2, \alpha_2, \beta_2) \quad (14)$$

$$where: \quad \gamma = (v_1 + v_2)^{-1}(v_1\gamma_1 + v_2\gamma_2), \quad v = v_1 + v_2 \quad (15)$$

$$\alpha = \alpha_1 + \alpha_2 + \frac{1}{2}; \beta = \beta_1 + \beta_2 + \frac{1}{2}v_1(\gamma_1 - \gamma)^2 + \frac{1}{2}v_2(\gamma_2 - \gamma)^2 \quad (16)$$

The NIG summation can effectively incorporate views of varying quality. Specifically, the parameter v can be interpreted as virtual evidence that indicates the confidence of an NIG distribution for the mean γ. As demonstrated in Eq. (15), if one view has greater confidence in its prediction, it will have a larger contribution to the final prediction. β directly reflects both aleatoric and epistemic uncertainty (Eq. (16)) and consists of two components: the sum of β_1 and β_2 from multiple views, and the variance between the final prediction and that of each individual view. Intuitively, the final uncertainty is jointly determined by the uncertainty specific to each view and the deviation in predictions among different views. We define the final loss function $\mathcal{L}_{all}(\boldsymbol{\theta})$ as the sum of losses of two views $\{\mathcal{L}_1(\boldsymbol{\theta}), \mathcal{L}_2(\boldsymbol{\theta})\}$ and the fused distribution $\mathcal{L}_f(\boldsymbol{\theta})$. The workflow of TrustGeo is shown in Figure 2: Firstly, the evidential uncertainties of the self-information view and graph learning view are learned separately. Then, two NIG distributions from different views are dynamically summed, to capture the view-specific uncertainty and generate a new NIG distribution for quantifying the holistic graph uncertainty. The proposed multi-task learning loss is backpropagated for model optimization.

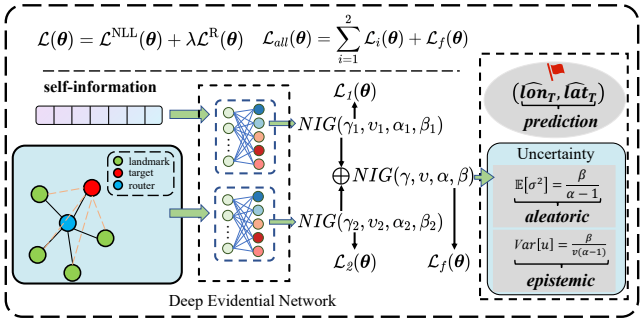


Figure 2: Illustration of the proposed TrustGeo.

4 EXPERIMENTS

We now present the setup of our extensive experiments, followed by the details of the results of different evaluations of TrustGeo.

Datasets. We use three large-scale real-world street-level IP geolocation datasets [34] collected from three metropolises, i.e., New York City, Los Angeles, and Shanghai, which consist of 91,808, 92,804, and 126,258 IP addresses, respectively. For the data processing, we randomly select 80% IPs for training and the remaining 20% for testing. In the training process, we take 70% IP as landmarks and 30% as target IPs following [34]. During testing, we treat the training set as landmarks and others as target IPs to report the results. 5-fold cross-validation procedure is utilized to mitigate the bias caused by the random data selection.

Baselines. We compared TrustGeo with the following state-of-the-art baselines, including *delay-based* measurement methods [4, 16, 26, 32], *attribute learning* methods [3, 14, 17, 36], and *graph learning* models [7, 34]: (1) *GeoPing* [26] is a network delay-based method that assigns the target IP the location of the closest landmark server in terms of the "nearest" neighbor in latency space. (2) *CBG* [10] establishes a continuous space and infers the geographic locations of Internet hosts using multilateration positioning with distance constraints. (3) *TBG* [16] is a topology-based geolocation approach

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

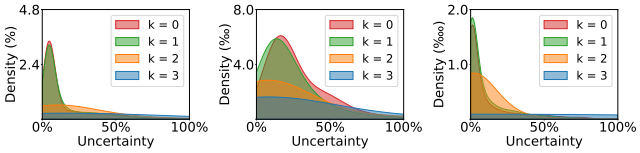
Type	Method	Time	New York			Los Angeles			Shanghai		
			RMSE	MAE	Median	RMSE	MAE	Median	RMSE	MAE	Median
Delay Measurement Methods	GeoPing	2001	14.25	11.66	10.24	18.19	13.40	9.450	29.23	24.84	22.23
	CBG	2006	18.74	14.89	11.97	21.05	17.77	13.93	34.60	28.77	24.28
	TBG	2006	16.20	13.15	10.57	17.98	15.60	12.32	27.24	22.53	19.97
	NCRGeo	2018	3.252	2.518	2.110	7.006	5.684	5.012	9.839	6.746	5.240
	XLBoost-Geo	2020	3.005	2.179	1.572	6.820	4.577	4.129	9.990	6.850	5.242
Attribute Learning Methods	NN-Geo	2016	7.740	6.369	5.654	11.19	9.322	8.672	10.25	7.178	5.148
	LightGBM	2017	3.356	2.963	2.827	6.968	5.878	5.119	9.924	6.431	4.898
	MLP-Geo	2020	7.127	5.858	5.077	10.54	8.651	7.567	10.03	7.017	5.107
	TabNet	2021	3.985	3.272	3.198	7.252	6.262	5.189	9.986	6.722	5.012
Graph Learning Methods	GNN-Geo	2021	3.014	2.135	1.618	6.807	4.655	4.039	9.645	6.026	4.482
	GraphGeo	2022	2.681	1.614	1.118	6.621	3.778	2.269	9.051	5.981	3.982
Ours	**TrustGeo**	-	**2.283**	**1.316**	**0.888**	**5.025**	**2.793**	**1.786**	**8.389**	**5.457**	**3.619**

that converts topology and communication delay into a set of constraints to geolocate routers and Internet hosts simultaneously. (4) *NCRGeo* [4] calculates the distance between the router and landmarks, and then uses multilateration to estimate geolocations. (5) *XLBoost-Geo* [32] extracts clues from web pages and associates the coordinate of the closest landmark from the common router surroundings with the location of the target IP. (6) *LightGBM* [17] is an advanced gradient-boosting framework that uses tree-based learning algorithms for geolocalization. (7) *TabNet* [3] is an automatic feature interaction learning model for tabular data, which exploits a self-attentive neural network for feature engineering and IP geolocation. (8) *NN-Geo* [14] collects RTTs from multiple observers as features, and uses a two-tier neural network that determines a rough region first and then narrows the scope of the target IP for street-level IP geolocation. (9) *MLP-Geo* [36] has a similar architecture with NN-Geo, but it uses router IDs between probing hosts and the targets as extra information. (10) *GNN-Geo* [7] is a framework that connects all landmarks to exploit surrounding knowledge for the target IP host. (11) *GraphGeo* [34] establishes neighborhood relationships between IP hosts from both topological and semantic perspectives. It also models network uncertainty via continuous normalizing flow.

Implementation Details. In the star GNN, both Agg() and Update() in Eq.(4) are implemented by a fully connected layer. The dimension $d_x + d_m$ of the node feature is set to 30 (14-dim for the attribute feature and 16-dim for the measurement feature). We split into training and validating sets (4:1) to optimize parameters. We tune all hyper-parameters systematically, using grid search for optimization within 2,000 epochs. For New York, Los Angeles and Shanghai datasets, the optimal learning rate is set to 0.005, 0.003, 0.0015, respectively. And the coefficient λ is set to 0.007, 0.007, 0.001, respectively. The learning rate is halved if there are no improvements after 5 epochs. The training process will halt when parameter updates no longer yield improvements for 50 epochs. We train with Adam optimizer [18] and report the average results of 20-time runs. We note that the results are statistically significant at the level of $p < 0.005$ using a paired t-test, and the experiments are conducted on `Intel i5-11400F 4.40GHz`, one `NVIDIA GeForce GTX 1080 Ti`, with 64GB CPU memory.

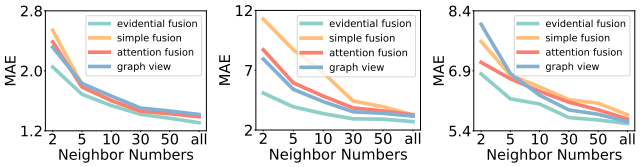
4.1 Performance and Robustness

The geolocation performance of TrustGeo and baselines is summarized in Table 1, from which we have the following observations:
(O1): Pure delay-based algorithms such as GeoPing and CBG, typically work poorly and generally cannot meet the requirement of the street-level IP geolocation task. Their performance is determined by the distance to the nearest landmarks and is thus poor due to the circuitousness problem and irregularity of Internet paths. To ensure performance, they require many landmarks that are carefully chosen to cover the target hosts. However, it is difficult to acquire landmarks uniformly distributed, resulting in an inferior estimation of the targets without landmarks coverage. In contrast, NCRGeo and XLBoost-Geo which combine the route information and/or other topology measurements often yield better performance.
(O2): Exploring rich attribute information can significantly improve the geolocation accuracy. Among attribute learning methods, NN-Geo and MLP-Geo perform worst as they only use RTTs and router IDs as features. LightGBM and TabNet explore more information (autonomous systems (AS) and WHOIS data), thus considerably reducing geolocation errors. These results suggest that external attribute information can help measurement-based models overcome insufficient structural constraints in the router-level graph.
(O3): Graph learning-based methods including our TrustGeo generally achieve the best geolocation performance. For example, the best attribute learning method LightGBM has around 3 km errors in New York City, but the worst graph learning method GNN-Geo can reduce the distance errors to ~2 km. This phenomenon can be explained by the principle of information entropy: exploiting neighbors' relationships has the opportunity to enrich the information regarding target hosts, and thus reduce the forecast uncertainty



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

Figure 3: Evidential robustness under measurement noise.



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

Figure 4: Influence of the number of neighbors (controlled by masking a proportion of positions in the adjacent matrix).

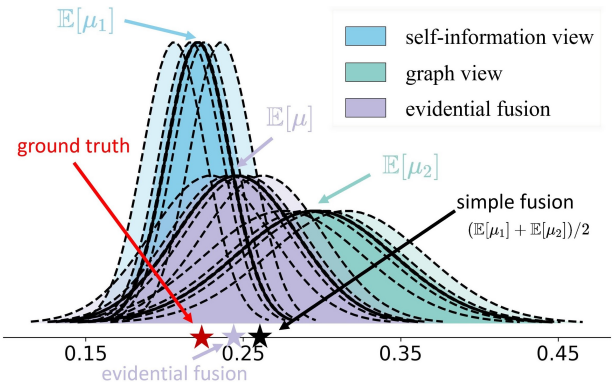


Figure 5: Evidential fusion. μ_1, μ_2, μ denote the mean value of self-information view, graph view, and evidential fusion.

while improving the geolocation accuracy. Notably, our star GNN-based approach can reduce the Median error to 0.888 km in the New York dataset, which is an encouraging result verifying our motivation in building star graphs.

To substantiate our claims that TrustGeo can respond to network environment variations, we visualize the distribution of uncertainty under simulated measurement noise in Figure 3. We consider the general case of measurement noise caused by network loss, latency, and congestion. We generate the simulated data to our test set using Gaussian noise perturbation since previous works [15, 24] have shown that measurement error on a network path approximately follows a normal distribution. k is set to control the intensity of random noise, i.e., $\epsilon \sim k\mathcal{N}(0, \sigma_m^2)$, where σ_m^2 equals to the magnitude of measurement values. Note that the purpose of this experiment is not to propose a defense for potential fluctuations, but rather to demonstrate that evidential models can accurately capture increased predictive uncertainty under unstable network scenarios. Figure 3 shows that even without adding additional simulated noise, the uncertainty score is smoothly distributed over the entire interval. Besides, the holistic uncertainty on three datasets follows: New York < Los Angeles < Shanghai, which is consistent with the prediction error in Table 1. The above phenomenon verifies that, despite the inherent measurement noise in our real-world dataset, evidential uncertainty learning is capable of capturing such statistics from the environment. Furthermore, as the noise level increases, the uncertainty of noisy samples also increases, which confirms that the proposed method can detect the intensity of noise in network, and make corresponding feedback in time through epistemic uncertainty.

4.2 Effectiveness of Evidential Fusion

We conduct an extra experiment to verify our motivation that the number of neighbors has a significant effect on graph learning performance, as well as verify the superiority of the evidential fusion strategy in TrustGeo. To this end, we control the number of neighbors by masking a proportion of nodes. We compare TrustGeo with other two common fusion strategies: (1) the *simple fusion strategy*, which simply uses the sum of decisions from all the views as the final decision; (2) the *attention fusion strategy*, which leverages the attention mechanism to fuse all the views together. We also compare TrustGeo with every single view. Note that we omit the self-information view in the above figures, since it will not change as neighbor numbers vary.

As shown in Figure 4, distance errors of all variants drop quickly when the number of neighbors increases from 0 to 30. It suggests that few neighbors are not sufficient for establishing the star GNNs to yield satisfying geolocation performance. Also, it verifies our former assumption that limited neighbor information in the dynamic IP graph causes the bottleneck and instability of graph learning. Furthermore, when comparing three information fusion strategies, the proposed evidential fusion learning outperforms the others, especially when there are only a few neighbors. Traditional information fusion strategies do not consider view-specific uncertainty and are thus unable to accommodate the variances of different views to different samples. In contrast, the evidential fusion strategy leverages the NIG summation operation to realize uncertainty-aware information fusion, adjusting the importance between inherent self-information and graph-learned information adaptively. Beyond trustworthy decisions, our approach automatically alleviates the impact of heavily noisy or corrupted views. In other words, compared with the deterministic fusion strategies, our method can effectively distinguish which view is noisy or corrupted for different samples, and accordingly can take the view-specific uncertainty into account for robust integration.

4.3 Case Study: Uncertainty Analysis

Evidential fusion strategy can improve the performance of the IP geolocation task and, to provide more insight, we visualize the intermediate fusion process to better understand the working principles. Concretely, we learn two distinct evidential parameter estimation models for the self-information view and graph view separately. These learned parameters are integrated to synthesize a new distribution, which is finally used to predict the target as well as

Table 2: Efficiency comparisons.

Method	MACs (10^6)	#Para. (10^3)	Training time (minute)		
			New York	Los Angeles	Shanghai
GNN-Geo	91.37	17.62	72.61	79.95	90.88
GraphGeo	12.76	255.2	51.55	58.82	67.52
TrustGeo	**0.544**	**5.632**	**17.61**	**17.97**	**27.48**

uncertainty from an input sample. Figure 5 illustrates that the evidential fusion strategy can alleviate the impact of heavily noisy or corrupted views. Compared to the simple fusion which uses the average mean value from two specific views, the prediction of evidential fusion is much closer to the ground truth.

4.4 Model Efficiency

We evaluate the model efficiency by comparing TrustGeo with: (1) graph learning methods (*GNN-Geo*, *GraphGeo*), and (2) uncertainty estimation methods (*Dropout* [9], *Ensemble* [20]).

The efficiency of the router-level graph. Table 2 summarizes the efficiency comparisons among three graph-based models in terms of multiply-accumulate operations per second (MACs), the number of parameters (#Para.), and training time. Due to the Memory Access Cost and data scale, the lower the MACs, the better the model's efficiency.Remarkably, TrustGeo runs ~3 and ~4 times faster than GraphGeo and GNN-Geo, respectively. Besides, TrustGeo only has 5.632K parameters – significantly fewer than GraphGeo and GNN-Geo. GNN-Geo establishes a complete geographic graph that contains all hosts and routers, which is hundreds to thousands of times bigger than the router-level graph and therefore requires significantly more computation time and memory. GraphGeo employs the continuous normalizing flow to approximate the network latency and congestion, which requires a sequence of invertible transformations that are computationally intensive. In comparison, TrustGeo only builds small-scale star graphs and does not need to do complex probabilistic transformations.

The efficiency of evidential uncertainty learning. Due to the intractability of the posterior in Bayesian probabilistic learning, existing works use sampling techniques to approximate it. Two classical and preeminent uncertainty estimation methods dubbed *Dropout* and *Ensemble* are used for comparison. Despite the significant performance on uncertainty estimation, they have a few notable downsides/limitations: – both *Dropout* and *Ensemble* are very slow since they require running the model multiple times. For *Ensemble*, it is even worse because it needs to initialize and train multiple independent models, which is extremely computationally costly. – Additionally, both algorithms impose a memory constraint since we need to keep all these models in parallel, which is a significant limitation for applications where uncertainty estimation is necessary to be made in real-time on edge devices, e.g., on mobile devices. Alternatively, evidential uncertainty learning is much more efficient since models can estimate uncertainty with only one pass.

4.5 Parameter Sensitivity

Parameter sensitivity of regularization coefficients. Figure 6 shows the importance of augmenting the training objective with

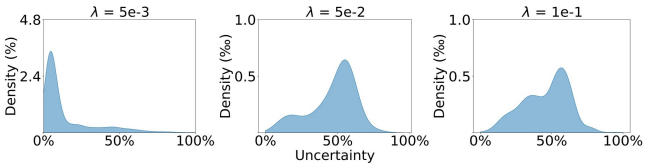


Figure 6: Regularization strength and epistemic uncertainty. We use the New York dataset as a study case.

the evidential regularizer $\mathcal{L}^R$, via quantitative results on epistemic uncertainty estimation after training with different realizations of the regularization coefficients λ. The ability to calibrate uncertainty on noisy data is heavily related to the intensity of the regularizer: – uncertainty decays to zero as the regularizer weight decreases; – stronger regularization inflates uncertainty. In sum, a careful task-specific choice of the hyper-parameter λ is needed.

5 CONCLUDING REMARKS

We presented TrustGeo, a novel framework for street-level IP geolocation. We proposed a novel router-level star GNN that exploits abundant information from adjacent neighbors to prevent the model from inaccurate relative distance estimation. Highlighting the importance of trustworthiness for establishing a geolocation system, we utilized evidential uncertainty learning to enable uncertainty estimation. Considering the dynamic property of the constructed router-level graph, we further introduced NIG summation operation for adaptive information fusion. Comprehensive experimental analysis on three real-world datasets verified the effectiveness and efficiency of TrustGeo.

Remarks. IP geolocation has several potential impacts on society: (1) Privacy: it can track individuals' online activities and locations. (2) Targeted advertising: it helps deliver personalized ads based on location. (3) Geo-restrictions: it enforces geographical restrictions on content. (4) Cybersecurity: it identifies and blocks malicious IP addresses. In a nutshell, it's important to use IP geolocation responsibly.

Uncertainty estimation for GNNs has a significant industrial impact. Although we considered a specific Web measurement scenario (IP geolocation), the proposed method can be extended to a range of web applications since network noise is prevalent and modeling uncertainty is important in both Web measurements and downstream applications. Cold-start is also common in many Web applications (e.g., recommendation and user modeling) involving automated data modeling. Our uncertainty-aware information fusion strategy adaptively evaluates the dynamic graph's quality while adjusting the coefficient weight to keep the holistic stability of the Web system. In our future work, we plan to incorporate the concept of adversarial training and causal inference into TrustGeo, towards building a more reliable and stable geolocation system.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (Grant No. 62176043 and 62072077), Natural Science Foundation of Sichuan Province (Grant No. 2022NSFSC0505), National Science Foundation SWIFT (Grant No. 2030249), and Hong Kong RGC TRS T41-603/20-R.

REFERENCES

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76 (2021), 243–297.

[2] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. 2020. Deep evidential regression. *Advances in Neural Information Processing Systems* 33 (2020), 14927–14937.

[3] Sercan Ö. Arik and Tomas Pfister. 2021. TabNet: Attentive Interpretable Tabular Learning. In *AAAI*. 6679–6687.

[4] Jing-ning Chen, Fen-lin Liu, Ya-feng Shi, and Xiangyang Luo. 2018. Towards IP location estimation using the nearest common router. *Journal of Internet Technology* 19, 7 (2018), 2097–2110.

[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] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural safety* 31, 2 (2009), 105–112.

[7] Shichang Ding, Fan Zhang, Xiangyang Luo, and Fenlin Liu. 2022. GNN-Geo: A Graph Neural Network-based Fine-grained IP Geolocation Framework. *arXiv preprint arXiv:2112.10767* (2022).

[8] Ziqian Dong, Rohan DW Perera, Rajarathnam Chandramouli, and KP Subbalakshmi. 2012. Network measurement based modeling and optimization for IP geolocation. *Computer Networks* 56, 1 (2012), 85–98.

[9] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.

[10] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. 2006. Constraint-based geolocation of internet hosts. *IEEE/ACM Transactions On Networking* 14, 6 (2006), 1219–1232.

[11] Chuanxiong Guo, Yunxin Liu, Wenchao Shen, Helen J Wang, Qing Yu, and Yongguang Zhang. 2009. Mining the web and the internet for accurate ip address geolocations. In *INFOCOM*. 2841–2845.

[12] Haosheng Huang, Georg Gartner, Jukka Matthias Krisp, Martin Raubal, and Nico Van de Weghe. 2018. Location based services: ongoing evolution and research agenda. *J. Locat. Based Serv.* 12, 2 (2018), 63–93.

[13] Hongbo Jiang, Jie Li, Ping Zhao, Fanzi Zeng, Zhu Xiao, and Arun Iyengar. 2021. Location Privacy-preserving Mechanisms in Location-based Services: A Comprehensive Survey. *ACM Comput. Surv.* 54, 1 (2021).

[14] Hao Jiang, Yaoqing Liu, and Jeanna N Matthews. 2009. IP geolocation estimation using neural networks with stable landmarks. In *INFOCOM Workshops*. 170–175.

[15] Mansour J Karam and Fouad A Tobagi. 2002. Analysis of delay and delay jitter of voice traffic in the Internet. *Computer Networks* 40, 6 (2002), 711–726.

[16] 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.

[17] 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.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Sándor Laki, Péter Mátray, Péter Hága, Tamás Sebők, István Csabai, and Gábor Vattay. 2011. Spotter: A model based active geolocation service. In *IEEE INFOCOM*.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems* 30 (2017).

[21] 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.

[22] Huan Ma, Zongbo Han, Changqing Zhang, Huazhu Fu, Joey Tianyi Zhou, and Qinghua Hu. 2021. Trustworthy Multimodal Regression with Mixture of Normal-inverse Gamma Distributions. *Advances in Neural Information Processing Systems* 34 (2021), 6881–6893.

[23] Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems* 31 (2018).

[24] Sue B Moon, Jim Kurose, and Don Towsley. 1998. Packet audio playout delay adjustment: performance bounds and algorithms. *Multimedia systems* 6, 1 (1998), 17–28.

[25] David Moore, Ram Periakaruppan, Jim Donohoe, and Kimberly Claffy. 2000. Where in the world is netgeo.caida.org?. In *INET*.

[26] Venkata N Padmanabhan and Lakshminarayanan Subramanian. 2001. An investigation of geographic mapping techniques for Internet hosts. In *SIGOMM*. 173–185.

[27] Hang Qian. 2018. Big data Bayesian linear regression and variable selection by normal-inverse-gamma summation. *Bayesian Analysis* 13, 4 (2018), 1011–1035.

[28] Jiaqian Ren, Lei Jiang, Hao Peng, Zhiwei Liu, Jia Wu, and S Yu Philip. 2022. Evidential Temporal-aware Graph-based Social Event Detection via Dempster-Shafer Theory. In *2022 IEEE International Conference on Web Services (ICWS)*. IEEE, 331–336.

[29] P. Sapiezynski, A. Stopczynski, R. Gatej, and S. Lehmann. 2015. Tracking Human Mobility Using WiFi Signals. *PLoS ONE* 10, 7 (2015).

[30] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. *Advances in neural information processing systems* 29 (2016).

[31] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. 2011. Towards street-Level client-Independent IP geolocation. In *NSDI*.

[32] 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 preprint arXiv:2010.13396* (2020).

[33] 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.

[34] Zhiyuan Wang, Fan Zhou, Wenxuan Zeng, Goce Trajcevski, Xiao Chunjing, Wang Yong, and Chen Kai. 2022. Connecting the Hosts: Street-Level IP Geolocation with Graph Neural Networks. In *SIGKDD*.

[35] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[36] Fan Zhang, Fenlin Liu, and Xiangyang Luo. 2020. Geolocation of covert communication entity on the Internet for post-steganalysis. *EURASIP Journal on Image and Video Processing* 2020, 1 (2020), 1–10.

[37] 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. *Computer Networks* 47, 4 (2005), 503–523.