Domain Adversarial Graph Convolutional Network Based on RSSI and Crowdsensing for Indoor Localization

Mingxin Zhang, Zipei Fan, Member, IEEE, Ryosuke Shibasaki, Member, IEEE, and Xuan Song, Member, IEEE

Abstract—In recent years, the use of WiFi fingerprints for indoor positioning has grown in popularity, largely due to the widespread availability of WiFi and the proliferation of mobile communication devices. However, many existing methods for constructing fingerprint data sets rely on labor-intensive and time-consuming processes of collecting large amounts of data. Additionally, these methods often focus on ideal laboratory environments, rather than considering the practical challenges of large multifloor buildings. To address these issues, we present a novel WiFi domain adversarial graph convolutional network model that can be trained using a small number of labeled site survey data and large amounts of unlabeled crowdsensed WiFi fingerprints. By constructing heterogeneous graphs based on received signal strength indicators (RSSIs) between waypoints and WiFi access points (APs), our model is able to effectively capture the topological structure of the data. We also incorporate graph convolutional networks (GCNs) to extract graph-level embeddings, a feature that has been largely overlooked in previous WiFi indoor localization studies. To deal with the challenges of large amounts of unlabeled data and multiple data domains, we employ a semisupervised domain adversarial training scheme to effectively utilize unlabeled data and align the data distributions across domains. Our system is evaluated using a public indoor localization data set that includes multiple buildings, and the results show that it performs competitively in terms of localization accuracy in large buildings.

Index Terms—Domain adaptation, graph neural network, indoor localization.

I. INTRODUCTION

Accurate indoor localization technology has the potential to support a variety of smart applications that involve sensing the environment and improving user experiences. While GPS is the dominant technology for outdoor localization, it tends to be less accurate in indoor environments due to the lack of outdoor exposure. As a result, researchers have been exploring various techniques to develop reliable indoor location systems. In recent years, using the existing WiFi infrastructure to build a localization system has emerged as a cost-effective and convenient solution, given the widespread availability of indoor WiFi signals [1], [2], [3]. One approach that has gained particular attention is the use of received signal strength indication (RSSI) to build WiFi fingerprints, as it is easily obtainable from ubiquitous smartphone devices and does not require the use of indicators like channel state information (CSI) [4].

Collecting a reliable data set of WiFi signals through a site survey can be a labor-intensive and time-consuming process, particularly in large multilevel buildings. Although there have been some studies working on using labeled or unlabeled large data set collected from users’ devices [5], this also poses the risk of label contamination and the difficulty of utilizing unlabeled data. In addition, RSSIs can be affected by reflection and diffraction constantly happening in the indoor environment, becoming susceptible, particularly in large buildings. A more complex problem is that, in a multilevel building, signals from an access point (AP) can be received by devices on multiple floors, this may cause a single 2-D coordinate label corresponding to multiple different RSSI features. In this article, we propose two models for indoor localization using WiFi signals: a semisupervised WiFi-domain adversarial graph convolutional network (WiDAGCN) model and a supervised WiFi attention graph convolutional network (WiAGCN) model, both based on graph convolutional network (GCN) [6]. To reduce the workload of data collection, our system utilizes crowdsensed data collected from user devices without the need for manual labeling. Because it is difficult to train a model with a small labeled data set, it is necessary to apply the unlabeled crowdsensing data to the training process as well. Based on the semisupervised adversarial training process, we conduct domain adaptation to align the data distribution of different data domains, making full use of unlabeled data and enhancing the robustness of the model.

To preserve the information contained in the data, we create topology graphs based on the connections between APs and waypoints as the input feature, rather than using fingerprint vectors directly. This approach not only reduces the reliance on labels, but also allows the model to learn from the higher order neighbors and improve its learning ability. Since seldom studies focus on the importance of the permutation invariance [7] in indoor localization problems, we also consider the importance of permutation invariance in indoor localization problems, and demonstrate the effectiveness of permutation-invariant models in improving accuracy. The GCN model is known to have permutation invariance...
properties, making it a suitable choice for building a graph-based permutation-invariant model [8], [9]. Also, GCN can extract graph-level embeddings, which allows us to obtain complex relationships between data points that may not be easily captured by other types of networks.

The structure of our indoor localization system is shown in Fig. 1. A small amount of labeled data is collected by site surveyors while a large unlabeled crowdsensing data set is also built. This allows us to build a sufficiently rich and usable data set with very little human effort. The model is trained using graphs from known environments, along with graphs from the new environment, allowing it to align the data distributions, improve robustness, and apply the knowledge learned from known environments to the new environment. The main contributions are summarized as follows.

1) Our models are designed to be able to be trained with very few labeled data, meaning that large-scale site surveys are not necessary. Instead, unlabeled data collected from users can be utilized to aid the training process through the use of semisupervised learning.

2) To make use of both labeled and unlabeled data, we present a new indoor localization system based on a semisupervised GCN.

3) We test the performance of different models, and the result shows that permutation-invariant features can indeed improve the accuracy of the indoor localization problem. We aggregate the whole subgraph features to get graph-level permutation-invariant representations.

4) Previous graph-based approaches have not typically considered the application of indoor localization in complex real-world scenarios. Our model is able to achieve room-level positioning accuracy in large-scale multi-floor buildings using only a small amount of labeled data.

The remainder of this article is organized as follows. We introduce some related work in Section II, then we define the problem and describe the data set preparation in Section III and present the details of the model design and training method in Section IV. We set up the experiment and evaluate our models in Section V. Finally, we discuss and conclude the work in Section VI and Section VII.
II. RELATED WORK

In this section, we provide a comprehensive review of various indoor localization technologies, with a specific focus on the localization system based on WiFi. We will also present some machine learning-based methods for WiFi localization and describe the application of GCN models in this context.

A. Indoor Localization System

In recent years, researchers have been actively investigating various technologies for indoor localization systems, with a focus on improving accuracy and robustness. In a previous study, the SnapLoc system, which leverages ultra-wideband (UWB) technology to estimate positions using the time-difference-of-arrival signals between anchors, was developed [10]. Other studies have focused on using embedded sensors on mobile phones, like the cellular, to provide localization services, such as the deep learning-based CellinDeep [11]. Among the various options, WiFi-based localization methods have become increasingly popular due to their widespread availability. Different indicators, such as CSI [4] and mmWave WiFi [12], have been used, with RSSIs being particularly easily obtainable from ubiquitous smartphone devices.

Since the pioneering study of RADAR [13], many studies have explored indoor localization methods using RSSI as location fingerprint [2], [3], [14], [15], [16]. With the development of machine learning technology, there are many novel ideas to solve the indoor localization problem.

B. Machine Learning-Based WiFi Localization

Xue et al. [17] proposed a weighted algorithm based on the physical distance of the RSSI and compared its performance with various $K$-nearest neighbors (KNN) models. The model was tested on two floors of a building and achieved an accuracy of 5.03 m. Niu et al. [14] proposed the WicLoc and achieved an average localization error of 4.60 m in a single environment. They also used different size of the data set to show the robustness of the model. However, their experimental environments were relatively ideal compared to real-world conditions. Traditional machine learning methods have faced challenges in improving performance further due to the high-dimensional data structure and complexity of indoor environments. Therefore, deep learning methods may be more suitable for addressing these types of problems [12].

As early as 2002, researchers have been exploring the use of multilayer fully connected neural networks for WiFi-based indoor localization using radio frequency (RF) data [18]. Since then, many other types of structures have been considered in an effort to extract richer features hidden in the data and improve performance in more complex experimental designs. Some stacked autoencoder (SAE) models based on fully connected networks have been tested, which can be trained in an unsupervised manner and then combined with a classifier for supervised training to create a localization model [19]. The use of convolutional neural networks (CNNs) as feature extractors has also been widely explored, with WiFi fingerprint vectors being input directly into 1-D CNNs [20] or transformed into 2-D format and processed using 2-D CNNs, similar to the way image data is commonly processed [21]. Some other researchers proposed models from the aspect of analyzing the trajectory or the moving pattern of the users. They naturally used long-short term memory (LSTM) to get the temporal features and improved the performance with the help of the time series [22]. While different models have varied structures, most WiFi indoor localization systems in recent years have primarily used fingerprint vectors composed of RSSI data as model inputs. While fingerprint vectors can be easily built, they do have limitations.

C. Graph Convolutional Network

To preserve more complete information and features in the input data, it is worth considering other data structures that can better represent the relationship between way-points and APs. In contrast to fingerprint vectors, graphs can represent the complex topological relationships between different entities, making the learning of features and information hidden within graphs a focus of research. GCN [6] can effectively learn and utilize features hidden in graphs through the use of spatial or spectral convolution operations. Veličković et al. [23] and Song et al. [24] modeled dynamic online community user behaviors with a recurrent neural network (RNN) and context-dependent social influence with a graph-attention neural network (GAT). Han and Xu [25], Cherian et al. [26] proposed a graph convolution collaborative filtering model based on the attention model and obtained more excellent feature using attention model to highlight the relationship and interaction information. While GCN can express the deep graph feature, it also guarantees the permutation invariance of the model [8], [9]. There also have been studies working on the network model optimization, for example, Chiang et al. [27] proposed Cluster-GCN made it easier to train a large-scale GCN. Researchers have constructed graphs for different ubiquitous computing scenarios, such as simulating the physical network topology [28], retaining the spatiotemporal information [29], [30].

There have been a number of studies that have implemented graph-based localization and mobility prediction systems [30], [31], but to the best of our knowledge, only a few studies have focused on using graph neural networks to build WiFi indoor localization systems [32], [33].

In summary, current research on RSSI-based indoor positioning tends to focus on laboratory or simple and ideal environments, while solutions suitable for complex, large-scale application scenarios often require labor-intensive and time-consuming data collection efforts. Furthermore, GCNs are well-suited for preserving topological feature information and the permutation invariance of sets. However, there is a lack of research on the use of GCNs for indoor localization.

III. PRELIMINARIES

In this section, we briefly describe the indoor localization problem and we introduce the public data set used in this article and the data preprocessing method.
A. Problem Definition

The RSSI fingerprint strengths of different APs are described by the RSSI fingerprint corresponding to each waypoint. For each waypoint, let $n$ be the number of selected APs in the environment. Then, we can obtain an RSSI vector $x_{\text{RSSI}} = [\text{RSSI}_1, \ldots, \text{RSSI}_n]$, where RSSI$_i$ is the RSSI value received from the $i$th AP. Similarly, let $bssid_i = [\text{bssid}_1, \ldots, \text{bssid}_n]$, where bssid$_i$ is the basic service set identifier (bssid) of the $i$th AP. Because only a few site surveys were conducted in our experiment scenario, there are very few labeled data in the target domain, which means we can only get a large number of crowdsensing data $x_{\text{RSSI}}$ and $bssid_i$ without corresponding coordinates. In addition, the equipment and arrangement of APs in different buildings are completely different. As a result, the focus of this article is on methods that make full use of both labeled and unlabeled data, helping the model fully learn the target-domain data distribution. We also aim to extract domain-invariant features and transfer knowledge learned in known source domains to the unknown target domain using cross-domain features.

B. Data Set Preparation

In this article, the Microsoft Indoor Location Competition 2.0 data set [34] is used. Five buildings are selected as pretraining sites, which contain known data to pretrain the model, and one building is designated as the target site, where the main focus of domain adaptation lies. The details of the domain setup and data set split are described in Section V-A. To filter out low-quality signal connections, only the 50 connections with the highest RSSI strength are considered for each waypoint in this article. The RSSI values are standardized by $\text{RSSI} = (\text{RSSI}_{\text{origin}} - \mu) / \sigma$, where RSSI$_{\text{origin}}$ represents the original values of RSSI, $\mu$ and $\sigma$ represent the mean and standard deviation of RSSI$_{\text{origin}}$, respectively, and RSSI represents standardized values. The bssids of APs are also transformed into consecutive integer indices, from 0 to $n_{\text{bssid}} - 1$.

IV. METHODOLOGY

In this section, we will describe the graph structure and design of our proposed deep learning model, which serves as domain adaptation. And different training schemes will also be introduced.

A. Graph Construction

Through signal propagation between APs and devices, we can naturally obtain the connection relationship between APs and devices, thus establishing heterogeneous graphs. We represent the AP nodes as $V_{\text{AP}}$ and the waypoint nodes as $V_{\text{WP}}$. In this way, there are also two types of relationships between nodes, $E_{\text{AP} \rightarrow \text{WP}}$ and $E_{\text{WP} \rightarrow \text{AP}}$, representing the edges from APs to waypoints and from waypoints to APs, respectively. Therefore, the node features can be represented as $F_{\text{AP}} = \{f_{1}^{\text{AP}}, f_{2}^{\text{AP}}, \ldots\}$ and $F_{\text{WP}} = \{f_{1}^{\text{WP}}, f_{2}^{\text{WP}}, \ldots\}$, the edge features can be represented as $F_{\text{AP} \rightarrow \text{WP}} = \{f_{1}^{\text{WP} \rightarrow \text{AP}}, f_{2}^{\text{WP} \rightarrow \text{AP}}, \ldots\}$ and $F_{\text{WP} \rightarrow \text{AP}} = \{f_{1}^{\text{AP} \rightarrow \text{WP}}, f_{2}^{\text{AP} \rightarrow \text{WP}}, \ldots\}$, where $f_{i}^{\text{AP}}$ represents the feature of the $i$th AP, corresponding to the node $v_{i}^{\text{AP}}$. In this article, we use the encoded bssids as the feature of AP nodes, which means that $f_{i}^{\text{AP}} = \text{bssid}_i$. And $f_{i}^{\text{WP}}$ is set as $(x_i, y_i)$, the 2-D coordinate of the $i$th waypoint, corresponding to $v_{i}^{\text{WP}}$. For the edges in the graph, they represent the relationship between APs and waypoints, so we can set $f_{i}^{\text{WP} \rightarrow \text{AP}}$ as the RSSI value between the starting point and the end point of the $i$th edge $e_{i}^{\text{WP} \rightarrow \text{AP}}$. And the condition of $F_{\text{WP} \rightarrow \text{AP}}$ is the same. In this way, we define the heterogeneous graph as $G = (V, E, F_{V}, F_{E})$, where $V = \{V_{\text{WP}}, V_{\text{AP}}\}$, $E = \{E_{\text{WP} \rightarrow \text{AP}}, E_{\text{AP} \rightarrow \text{WP}}\}$, $F_{V} = \{F_{\text{WP}}, F_{\text{AP}}\}$ and $F_{E} = \{F_{\text{WP} \rightarrow \text{AP}}, F_{\text{AP} \rightarrow \text{WP}}\}$ represent the nodes, edges, and the features in the graph, respectively.

To construct a large graph, we use all known nodes and edges. Inspired by the concept of Deep Sets, we use the aggregated features of the node to be predicted and its surrounding nodes rather than the features of a single node. The steps for constructing a subgraph are shown in Fig. 2. A subgraph corresponding to each waypoint is created by traversing all waypoints.

In the case of the source domain, the feature is also the label of the waypoint, meaning that the features of the waypoints to be predicted are unknown. Therefore, we set an all-ones vector as the feature of the waypoint to be predicted (yellow node in Fig. 2) and mark the waypoint to be predicted as a new type of node. We also create a pair of new edge types between the waypoint to be predicted and the AP (WP to predict $\rightarrow$ AP, AP $\rightarrow$ WP to predict). Next, all 50 neighbors of the waypoint to predict are selected and added to the
subgraph. We randomly select five 2nd-order neighbors for each 1st-order neighbor and add them into the subgraph. There will be 2000 to 3000 nodes in a single subgraph if we use all 1st-order and 2nd-order neighbors, make the model difficult to train. Therefore, we use the method of multiple repeat sampling to reduce the complexity of the graph and to retain as much information as possible in the graph. We conduct five times repetitive sampling so there are five subgraphs with 301 each 1st-order neighbor and add them into the subgraph. There

Because 50 WiFi signal connections are kept for each (x, y) coordinate, there are 50 bssids and 50 corresponding RSSIs. The original bssids and RSSIs are processed by the embedding layer and the fully connected encoding layer, respectively, transformed into 10 × 50 feature matrices \( F_{\text{bssid}} \) and \( F_{\text{RSSI}} \). Then, these two matrices are concatenated into a 20 × 50 matrix, representing the fingerprint feature set. After the \( \phi \) network, the matrix is added along the axis that represents the set elements, resulting in a vector representing the features of the set. Finally, the \( \rho \) network gives the predicted coordinate of the waypoint with the vector. To verify if the model retains the permutation-invariant feature of the set and has a better performance, a fully connected neural network with the same structure is constructed as a control group.

2) WiDAGCN Model: Inspired by the idea of the Deep Sets, we propose a WiDAGCN. While GCN can express the deep graph feature, it also guarantees the permutation invariance of the model [8], [9]. But first, before the description of WiDAGCN model, a supervised model that has a similar structure will be introduced.

This supervised WiAGCN model consists of signal encoding layers, GCN layers, and a GAT [23] layer. The structure of WiAGCN is shown in Fig. 5.

As mentioned in Section IV-A, the input subgraph contains three types of nodes and four types of edges. Similar to the Deep Sets model, an embedding layer is used to represent the bssid features, and other nodes and edge features are represented by full-connected layers. Two weighted heterogeneous graph convolutional layers are used to obtain the node embedding. With the GAT layer, we can get multiple node feature representations with the multihead attention mechanism. The key point to maintaining the permutation-invariant feature is the readout function, which is similar to the permutation-invariant operation in Deep Sets model. We aggregate the mean node representation of three types of nodes and get a graph-level feature. Then, we aggregate the central waypoint (the waypoint to predict) and the surrounding 1st-order neighbors, the APs, to get a local feature. Finally, we extract the feature of the center point. These three different level features are concatenated to obtain the final regression result.

In this study, we use deep graph library (DGL) [35] to implement the graph-based model. The principles of the

The idea of Deep Sets network is using the deep network to fit the function \( \phi \) and \( \rho \). The structure of the Deep Sets model implemented in this article is shown in Fig. 4.

As shown in Fig. 4, elements of the fingerprint set are RSSI/bssid composite features corresponding to each available AP. For each waypoint, the model can be described as follows:

\[
X = \begin{bmatrix} F_{\text{RSSI}} & F_{\text{bssid}} \\ \text{RSSI}_0 & \text{bssid}_0 \\ \text{RSSI}_1 & \text{bssid}_1 \\ \vdots & \vdots \\ \text{RSSI}_{49} & \text{bssid}_{49} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{49} \end{bmatrix}
\]

\[
\text{output} = \rho(\sum_{x \in X} \phi(x)) = \rho(\phi(x_0) + \phi(x_1) + \cdots + \phi(x_{49})).
\]

\[
(1)
\]

(2)

\[ (1) \]

\[ (2) \]
graph neural network model used in DGL will be described next.

Because we use connections between APs and waypoints as edges and RSSI values as edge features, we should consider the weighted graph convolution to utilize the edge information. The weighted graph convolution used in DGL GraphConv module was introduced in the study [6] and can be mathematically defined as

\[
h_{l+1}(i) = \sigma \left( h_l(i) + \sum_{j \in N(i)} e_{ij} h_l(j) W_l(i) \right)
\]

where \(N(i)\) is the set of neighbors of node \(i\), \(c_{ij}\) is the product of the square root of node degrees, the superscript \(l\) or \(l+1\) represent the \(l\)th layer or the \(l+1\)th layer, \(\sigma\) is an activation function, \(h_l(i)\) represents the input, \(h_{l+1}(i)\) represents the output, \(W_l\) represents the weight matrix, and \(e_{ij}\) is the scalar weight on the edge from node \(j\) to node \(i\).

All nodes in the neighborhood share the same convolution kernel parameters, and the performance of the model may be limited. To differentiate between nodes of different importance in the neighborhood, we can apply the attention mechanism to the graph convolution model using GAT layers, so that neighbor nodes can be given different weights and importance. DGL also gives the implementation of the GAT layer [23], and the mathematical definition can be described as

\[
h_{l+1}(i) = \sum_{j \in N(i)} \alpha_{ij} W_l(i) h_{l}(j).
\]

The attention score between node \(i\) and node \(j\) is represented as \(\alpha_{ij}\)

\[
\alpha_{ij} = \text{softmax}\left( e_{ij} \right)
\]

\[
e_{ij} = \text{LeakyReLU}\left( \vec{a}^T \left[ W_l(i) || W_l(j) \right] \right)
\]

where \(\cdot^T\) represents transposition and \(||\) is the concatenation operation. The attention mechanism can be defined as a single-layer feedforward neural network and can be parameterized by the weight vector \(\vec{a}\).

In addition, because there are several different types of nodes and relations, we should also aggregate information from different parts of the graph together. Therefore, we use
the following method to perform the heterogeneous graph
convolution
\[
h^{(l+1)}_{dst} = \text{AVG}_{r \in R, r_{dst}=dst} \left( f \left( g_r, h^{l}_{src}, h^{l}_{dst} \right) \right)
\] (6)
where \( f \) is GCN or GAT module corresponding to each relation \( r \), the subscripts \( src \) and \( dst \) represent the source and the destination of a relation \( r \), and \( g_r \) represents the graph containing \( r \). The aggregation method used to aggregate the results from different relations in this article is defined as average.

However, it is difficult for this WiAGCN model to use few labeled data to train a robust model. Therefore, we propose the WiDAGCN model, which can be trained by the semisupervised method, making full use of unlabeled data. The structure of the WiDAGCN model is shown in Fig. 5. The shallow layers of WiDAGCN are the same as the WiAGCN model so we can use the pretrained model as a reliable initial condition. For the sake of brevity, we call this same part of network \( D(x) \) a 2-layer regressor while making it difficult for the domain discriminator to identify the domains that the data come from. We build a 2-layer regressor \( R(x) \) and a 3-layer domain discriminator \( D(x) \) with gradient reversal layer (GRL) [36] for the adversarial model. With the help of GRL, the forward and backward propagation of the model is different
\[
\text{GRL}(x) = x, \quad \frac{d\text{GRL}}{dx} = -\alpha I
\] (7)
where GRL(x) represents the GRL function, \( I \) is the identity matrix, and \( \alpha \) is a constant.

C. Supervised Learning

For the Deep Sets model and the WiAGCN model, simple fine-tuning transfer is used. First, we use labeled source-domain data to train a pretrained model
\[
\arg\min_{\theta_G, \theta_R} \sum_{i=1}^{n} E_{(x,y) \in D_S} \left[ L_y(R(G(x)), y) \right].
\] (8)
Mean squared error (MSE) loss \( L_y \) is used in the coordinate regression problem. The \((x,y) \in D_S \) means data \( x \) and label \( y \) from the source domain \( D_S \), \( i = 0, 1, 2, \ldots \), and \( \theta \) represents the model parameters. Then, three different fine-tuning methods are compared
\[
\arg\min_{\theta_G} \sum_{i=1}^{n} E_{(x,y) \in D_T} \left[ L_y(R(G(x)), y) \right]
\] (9)
\[
\arg\min_{\theta_R} \sum_{i=1}^{n} E_{(x,y) \in D_T} \left[ L_y(R(G(x)), y) \right]
\] (10)
\[
\arg\min_{\theta_G, \theta_R} \sum_{i=1}^{n} E_{(x,y) \in D_T} \left[ L_y(R(G(x)), y) \right].
\] (11)
The \((x,y) \in D_T \) represents a small amount of labeled data \((x,y) \) in the target domain \( D_T \). Equations (9) and (10) show the \( G(x) \) fine-tuning and \( R(x) \) fine-tuning, respectively. For example, in (9) and (10), we tune the \( G(x) \) (\( R(x) \)) network and fix the parameters of the other part. For the Deep Sets model, due to the existence of the sum operation, the network is divided into two parts, the \( \phi \) network, and the \( \rho \) network. Similarly, we also fine-tune the \( \phi \) network and the \( \rho \) network separately. In addition, as shown in (11), we also perform an all-fine-tuning training, using this small amount of data to tune all the parameters of the model.

D. Semisupervised Adversarial Domain Adaptation

As introduced in Section IV-C, if we only have a very small amount of labeled target-domain data, it may be difficult or even unable to perform the fine-tuning process. Thus, the following training framework is used based on our problem. The goal is to learn a feature extractor network that produces domain-invariant features. Such features can be learned with the discriminator \( D(x) \) acting as an adversary. In this way, the generalized features can be aligned to the same distribution, no matter the data that come from which domain. The graph feature extractor \( G(x) \) and regressor \( R(x) \) are trained to make the \( R(x) \) get accurate positioning coordinates. The point is, that the feature extractor \( G(x) \) is also updated to make the discriminator \( D(x) \) can not give the correct domain classification results. This competing training goal can be described like
\[
\arg\min_{\theta_G, \theta_R} \sum_{i=1}^{n} E_{(x,y) \in D_S} \left[ L_y(R(G(x)), y) - L_d(D(G(x)), d_S) \right]
\]
\[
- E_{x \in D_T} \left[ L_d(D(G(x)), d_T) \right]
\] (12)
\[
\arg\min_{\theta_R} \sum_{i=1}^{n} E_{(x,y) \in D_S} \left[ L_d(D(G(x)), d_S) \right]
\]
\[
+ E_{x \in D_T} \left[ L_d(D(G(x)), d_T) \right]
\] (13)
where \( L_d \) is the cross-entropy loss used to calculate the loss of domain classification, \( x \in D_T \) represents the unlabeled data in the target domain, and \( d \) represents the domain label. With (12), the parameters of the feature extractor and regressor are updated so that the regressor can give accurate coordinates, while the discriminator cannot predict the correct domains. Equation (13) updates the discriminator to correctly predict the domain labels. Using the GRL, we can combine these two competing steps together
\[
\arg\min_{\theta_G, \theta_R, \theta_D} \sum_{i=1}^{n} E_{(x,y) \in D_S} \left[ L_y(R(G(x)), y) \right]
\]
\[
+ L_d(D(GRL(G(x))), d_S) \right]
\]
\[
+ E_{x \in D_T} \left[ L_d(D(GRL(G(x))), d_T) \right].
\] (14)

V. EXPERIMENT

The experiment environment in this article: The operating system is Ubuntu 18.04.5 LTS, the CPU is Intel Xeon Silver 4210 @ 2.20 GHz, and the GPU is GeForce RTX 2080 Ti.

A. Experiment Setup

In this article, we mainly focus on adaptation on different floors. Although floors in the same site may share many APs,
different floor plans and signal attenuation can confuse the model. As mentioned in Section III-B, site 5 is chosen as the target site, where our adaptation framework mainly works on. The details of the floor information are shown in Table I. And the floor plans are also shown in Fig. 6.

These five floors are set as the target domain in turn, while the other four floors are set as four source domains. The labels of source-domain data are all known, and some of the target-domain labels are known. We test the model on conditions of different sizes (1%, 5%, 10%, 20%, 30%, 40%) of training data with known labels. In the supervised learning setting, the labeled data in the target domain are used for training, and the remaining unlabeled data are divided into a validation set and a test set according to a 2:8 ratio. In the adversarial training setting, all data in the target domain participate in the training process, regardless of whether they are labeled. And the validation set and test set are also divided according to a ratio of 2:8 with the unlabeled data. Thus, the testing and validation process can also be viewed as using the model to label the unlabeled data. For the adversarial model, let these labeled waypoints as \( (x, y) \in D_T \) and the rest unlabeled fingerprints as \( x \in D_J^T \). The initial learning rates of \( G \), \( C \), and \( R \) are all set to 0.001, and if the validation loss does not decrease for ten consecutive epochs, the learning rate of \( G \) and \( R \) will be halved. All \( G \), \( C \), and \( R \) networks use Adam optimizer. For the GRL, the initial value of \( \alpha \) is set to 0, and the value of \( \alpha \) increases by 0.0001 with each epoch. The total number of the training epoch is set as 250 for each experiment. Ten times repeating experiments are performed for each floor, so the result of each model for each training data proportion is the average of total 50 times experiment results.

### Table I
**Detailed Floor Information of the Site 5**

| Floor | Width (m) | Length (m) | Waypoint Number |
|-------|-----------|------------|-----------------|
| B1    | 52.02     | 87.83      | 716             |
| F1    | 135.59    | 47.41      | 1631            |
| F2    | 135.70    | 47.79      | 1756            |
| F3    | 135.25    | 36.25      | 552             |
| F4    | 135.25    | 36.25      | 645             |

### Table II
**Average Errors (Unit: m) of Different Fine-Tuning Methods When Using 10% of the Labeled Training Data**

| Model      | If use pre-trained model | Fine-tuning Method | \( \phi \) or \( G \) | \( \rho \) or \( R \) |
|------------|--------------------------|--------------------|------------------------|-----------------------|
| Deep NN    | ✓                        | all-fine-tuning    | 15.72                  | 15.76                 | 21.74                |
| Deep Sets  | ×                        | \( \phi \)         | 8.22                   | 8.30                  | 7.70                 |
|            |                          | \( \rho \)         | 12.75                  | 12.74                 | 12.96                |
| GCN        | ✓                        | all-fine-tuning    | 6.95                   | 7.02                  | 7.41                 |
|            |                          | \( \phi \)         | 7.38                   | 7.44                  | 7.84                 |
| WiAGCN     | ✓                        | all-fine-tuning    | \textbf{6.58}          | \textbf{6.61}         | \textbf{7.00}        |
| (Proposed) | ×                        | \( \phi \)         | 6.70                   | 6.69                  | 7.42                 |

In Fig. 6, the structures of the floors in the target site vary from one another. Additionally, it is feasible for APs located on a single floor to be detected by wireless devices situated on multiple floors. Consequently, even though the RSSIs of a specific AP can be utilized for localization purposes across different floors, the propagation of WiFi signals is susceptible to alterations between floors due to the varying floor heights and the influences of walls, ceilings, and other environmental factors. By conducting the experimental design described above, we aim to allow the model to learn general knowledge across the different floors. This knowledge could be in the form of a mapping of data distribution in a high-dimensional space, where data from different domains can be aligned. Our goal is to achieve a more accurate indoor positioning system by leveraging this knowledge. This would involve learning how the WiFi signals are affected by the structure and other factors of each floor, and how they are transmitted between floors, so that the model can make more accurate predictions about a user’s location.

### B. Results

The total time for constructing the subgraph in the target site was 1926 s, or approximately 32 min. As there were five floors in the target site and five repetitions of sampling were conducted on each floor, the average time cost of constructing the subgraph on a single floor was approximately 77.04 s.
TABLE III
IMPACT OF LABELED TRAINING SET SIZE ON AVERAGE ERRORS (UNIT: M)

| Model        | Pretraining | Permutation Invariant Operation | Adversarial Component | Size of Labeled Training Dataset |
|--------------|-------------|---------------------------------|-----------------------|---------------------------------|
|              |             |                                 |                       | 1%      | 5%      | 10%     | 20%     | 30%     | 40%     |
| KNN          | ×           | ×                               | ×                     | 25.46   | 22.87   | 22.49   | 22.17   | 21.91   | 21.45   |
| Decision Tree| ×           | ×                               | ×                     | 24.84   | 16.41   | 14.63   | 12.51   | 11.42   | 11.06   |
| Deep NN      | ✓           | ×                               | ×                     | 21.45   | 17.64   | 15.72   | 13.74   | 12.71   | 11.93   |
| SAE          | ✓           | ×                               | ×                     | 51.00   | 21.67   | 16.26   | 13.14   | 12.15   | 11.10   |
| 1-D-CNN      | ✓           | ×                               | ×                     | 21.48   | 16.30   | 11.99   | 10.93   | 9.90    | 9.53    |
| Deep Sets    | ✓           | ✓                               | ×                     | 13.45   | 9.39    | 8.22    | 6.03    | 5.61    | 5.62    |
| DANN         | ✓           | ×                               | ✓                     | 19.47   | 11.28   | 8.91    | 6.53    | 5.36    | 5.09    |
| GCN          | ✓           | ✓                               | ×                     | 10.05   | 7.80    | 6.95    | 6.22    | 5.83    | 5.56    |
| WiAGCN (Proposed) | ✓       | ✓                               | ✓                     | 9.67    | 7.35    | 6.58    | 5.97    | 5.58    | 5.32    |
| WiDAGCN (Proposed) | ✓       | ✓                               | ✓                     | 9.09    | 7.59    | 6.04    | 5.67    | 5.20    | 4.84    |

In total, the 50 repeated experiments took approximately 100 h to complete, with each experiment taking approximately 2 h. During the experiment process, the GPU memory usage ranged from 1.6 GB to 5.0 GB. While the resources consumed during testing are of particular concern for the practical application of the model, our model was able to generate approximately 430 localization results in 11 s, with each localization process taking approximately 0.026 s. The GPU memory usage during testing was approximately 0.9 GB, including the test data set, models, and other consumption.

We first compare the performance of different fine-tuning methods of supervised models. As we mentioned previously, we fine-tune the $\rho$ or $G$ network, $\phi$ or $R$ network and the whole network, respectively. And we also compare the accuracy of models that use or do not use other sites to pretrain. The average positioning errors of 50 repeated experiments are shown in Table II. Because semisupervised WiDAGCN does not apply to the fine-tuning method, it is not concluded in Table II. The performance of WiDAGCN will be described in more detail later.

As mentioned in Section IV-B1, a Deep NN model consisting of only fully connected layers without permutation-invariant operation is built. Similarly, a GCN model without an attention mechanism and multilevel feature aggregation is also implemented. Despite having similar structures, the Deep NN model performed poorly compared to the Deep Sets network. By using the permutation-invariant operation, the positioning error was reduced by 47.70%, 47.33%, and 64.58% in the cases of all-fine-tuning, $\phi$ fine-tuning, and $\rho$ fine-tuning, respectively. The results showed that fine-tuning the deep layers of the Deep Sets model outperformed fine-tuning the shallow layers, and there was no significant difference between all-fine-tuning and $\phi$ network fine-tuning. This finding is consistent with the results presented by Yosinski et al. [37], which showed that shallow-layer features are applicable to different tasks and are commonly kept fixed in transfer learning while the deep layers are updated according to the new data set. Both of the tested graph-based models showed better performance with only a few dozen labeled data available. On the one hand, GCN has the property of maintaining a good performance even if data are sparse [6]. On the other hand, higher order neighbors are also considered so that more complex features can be utilized. However, the graph-based models showed some inconsistent results. Unlike Deep Sets, shallow layers fine-tuning performed better in terms of accuracy for GCN. The difference between all-fine-tuning and $G$ network (corresponding to $\phi$ network in Deep Sets) fine-tuning was also not significant. There has been limited research on fine-tuning GCN, but we can find some similarities in a study of CNN. Amiri et al. [38] found that shallow layers fine-tuning outperformed deep layers fine-tuning when using a small data set with U-Net. In our problem, this may be due to low-level graph patterns associated with shallow layers.

The results of using different sizes of the labeled training data set are shown in Table III. We have marked several operations conducted on compared models to conveniently compare the effect of different components of the model. To further validate the improvement of the adversarial component, we also built a domain-adversarial neural network (DANN) [36] with a similar structure to the Deep NN and Deep Sets models. In addition, we implemented two previous indoor localization systems, SAE [19] and 1-D CNN [20], in our environment. In the original study of SAE, the researchers trained the SAE in an unsupervised way like the common autoencoder [19]. The input and output were both the fingerprint vector, and each layer was trained independently to extract high-level features. Then, the encoder part was connected to a classifier and the decoder part was discarded. Finally, a supervised training process was conducted to obtain the localization model. We followed the steps and parameter setup of the original study and applied the model to our environment. We also implemented the 1-D CNN following the original study. The result shows that both the permutation-invariant operation and the adversarial component can help the model obtain a higher accuracy from the comparison between the Deep NN, Deep...
Sets and DANN, and both the Deep Sets and the DANN models were able to predict positions more accurately than the SAE and 1-D CNN models in an environment with limited labels. This demonstrates that both the permutation-invariant operation and the adversarial component can help improve the performance of the model. The DANN performed poorly when the number of labeled data was extremely limited, but its performance improved significantly as the number of labels increased. The results of the WiAGCN and the WiDAGCN models also showed that the adversarial component had a positive effect on the graph-based model. When 40% labeled data (220–702 pieces of data) are used, the model achieves the best improvement of 9.02% (from 5.32 to 4.84 m). The graph-based models obtain better accuracy when there are only 1% or 5% labels. Our proposed WiDAGCN model combines the advantages of above components and achieves the best results overall. WiDAGCN performed slightly worse than WiAGCN in the case of 5%, which could be related to the distribution of the data at the time of sampling.

To delve deeper into the performance of the proposed WiDAGCN model, additional experiments were conducted using models of varying scales. In the above experiments, the size of the GCN hidden layer in the WiDAGCN model was set to 32 for each layer. Subsequently, the performance of models with hidden layer sizes of 16 and 64 was evaluated, using the same set of hyperparameters, to analyze the impact of the scale of the model on performance. The results of these experiments are presented in Table IV. In terms of training time, there is no significant difference between the three models. Moreover, the difference in localization accuracy is also not noticeable, but the model with hidden layer size of 32 achieved better results. This is likely due to the fact that larger models still have not converged completely with the same hyperparameter settings, while smaller models are somewhat limited in their expressive capabilities. In this way, we can strike a balance between model complexity and expressiveness, using the simplest and most efficient models to address practical problems.

VI. DISCUSSION

In this study, our primary goal was to improve localization accuracy, taking into account the complexity and changes of the environment. In comparison to some existing studies, we focused on the approach of using fewer labeled data to address the problem in more complex application environments, particularly large-scale multifloor buildings. For example, the ViVi model [2] performed well in three test areas: an office building, an academic building, and a classroom building. However, tens of thousands of pieces of data were used for model training. The weighted algorithm [17] used only 2680 pieces of data, but the experiment environment was much simpler than the application environment. Our scenarios were set in a shopping mall with five floors, which is several times larger than other common experiment environments. In summary, we were able to achieve localization accuracy similar to existing studies using a small amount of labeled data in more complex, multi-environment settings.

There are still some issues that need to be addressed or that we have overlooked in this study. We believe that constructing a more rational graph structure is key to solving the problem. The construction of graphs for indoor positioning problems is a topic that merits further discussion, and how to construct a graph with a simple structure that better reflects the original information is also a topic we will investigate in the future. From a model design perspective, the subgraph regression approach used in this study focuses on the nodes and all parts within their second-order neighbors. In terms of maintaining the permutation invariance of features, the subgraph regression approach is more suitable for the Deep Sets model. However, in terms of the localization problem itself, it is more akin to a node regression problem, where the model needs to give the coordinate values corresponding to the nodes. The advantages and disadvantages of these two approaches need to be analyzed and compared further.

VII. CONCLUSION

In this article, we propose an RSSI-based GCN indoor localization model called WiDAGCN. We construct heterogeneous graphs based on the signal connections between users’ devices and APs in order to consider information from neighbors or higher order neighbors, or even unlabeled nodes. We design a domain adversarial GCN model to align the distribution of data from different environments. We focus on permutation-invariant features and obtain the aggregation representation of the graph. We test our model using a novel, real-world indoor localization public data set and find that it achieves competitive performance compared to state-of-the-art methods. This means that we can conduct simple site surveys and use unlabeled signal records received by user devices as auxiliary data to train a reliable positioning model.

REFERENCES

[1] H. Li, X. Chen, J. Wang, D. Wu, and X. Liu, “DAFI: WiFi-based device-free indoor localization via domain adaptation,” Proc. ACM Interact. Mobile Wearable Ubiquitous Technol., vol. 5, no. 4, pp. 1–21, 2021.
[2] C. Wu, J. Xu, Z. Yang, N. D. Lane, and Z. Yin, “Gain without pain: Accurate WiFi-based localization using fingerprint spatial gradient,” Proc. ACM Interact. Mobile Wearable Ubiquitous Technol., vol. 1, no. 2, pp. 1–19, 2017.
[3] J. E. van Engelen, J. J. van Lier, F. W. Takes, and H. Trautmann, “Accurate WiFi-based indoor positioning with continuous location sampling,” in Machine Learning and Knowledge Discovery in Databases, U. Brefeld et al., Eds. Cham, Switzerland: Springer Int., 2019, pp. 524–540.
[4] S. M. Hernandez and E. Bulut, “Performing WiFi sensing with off-the-shelf smartphones,” in Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops), 2020, pp. 1–3.
[5] H. Wang, S. Sen, A. Elghoary, M. Farid, M. Youssef, and R. R. Choudhury, “No need to war-drive: Unsupervised indoor localization,” in Proc. 10th Int. Conf. Mobile Syst. Appl. Serv., 2012, pp. 197–210.

[6] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), Toulon, France, Apr. 2017, pp. 1–14. [Online]. Available: OpenReview.net

[7] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, “Deep sets,” in Advances in Neural Information Processing Systems, vol. 30, I. Guyon et al., Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017.

[8] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, “Geoem-GCN: Geometric graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–12.

[9] I.-J. Liu, R. A. Yeh, and A. G. Schwlng, “PIC: Permutation invariant critic for multi-agent deep reinforcement learning,” in Proc. Conf. Robot Learn., vol. 100, Mar. 2020, pp. 590–602.

[10] B. Großwindhager, M. Stocker, M. Rath, C. A. Boano, and K. Römer, “A survey of deep learning approaches for WiFi-based indoor positioning,” J. Inf. Telecommun., vol. 6, no. 2, pp. 163–216, 2022.

[11] H. Rizk, M. Torki, and M. Youssef, “CellinDeep: Robust and accurate cellular-based indoor localization via deep learning,” IEEE Sensors J., vol. 19, no. 6, pp. 2305–2312, Mar. 2019.

[12] X. Feng, K. A. Nguyen, and Z. Luo, “Attention-based graph convolution collaborative filtering,” in Proc. 6th Int. Conf. Comput. Artif. Intell., 2020, pp. 58–62.

[13] J. Niu, B. Wang, L. Cheng, and J. J. P. C. Rodrigues, “WicLoc: An ultra-fast UWB-based indoor localization system for an unlimited number of tags,” in Proc. 18th Int. Conf. Inf. Process. Sens. Netw., 2019, pp. 61–72.

[14] J. Cherian, J. Luo, and S.-S. Ho, “ParkLoc: Light-weight graph-based vehicular localization in parking garages,” Proc. ACM Interact. Mobile Wearable Ubiquitous Technol., vol. 2, no. 3, pp. 1–23, Sep. 2018.

[15] Y. Yu, H. Wang, and Z. Li, “Inferring mobility relationship via graph embedding,” Proc. ACM Interact. Mobile Wearable Ubiquitous Technol., vol. 2, no. 3, pp. 1–21, Sep. 2018.

[16] X. Yang, S. He, B. Wang, and M. Tabatabaei, “Spatio-temporal graph attention embedding for joint crowd flow and transition predictions: A Wi-Fi-based mobility case study,” Proc. ACM Interact. Mobile Wearable Ubiquitous Technol., vol. 5, no. 4, pp. 1–24, Dec. 2022.

[17] M.-J. Chiu, Z. Liu, Y. Yin, A.-A. Liu, and R. Zimmermann, “Zero-shot multi-view indoor localization via graph location networks,” in Proc. 28th ACM Int. Conf. Multimedia, 2020, pp. 3431–3440.

[18] H. Zheng et al., “GraFin: An applicable graph-based fingerprinting approach for robust indoor localization,” in Proc. IEEE 27th Int. Conf.Parallel Distrib. Syst. (ICPADS), Dec. 2021, pp. 747–754.

[19] Z. Wu, X. Wu, and Y. Long, “Multi-level federated graph learning and self-attention based personalized Wi-Fi indoor fingerprint localization,” IEEE Commun. Lett., vol. 26, no. 8, pp. 1794–1798, Aug. 2022.

[20] Y. Shu, Q. Xu, J. Liu, R. R. Choudhury, N. Trigoni, and V. Bahl, “Indoor location competition 2.0 Dataset,” Jan. 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/indoor-location-competition-2-0-dataset/

[21] M. Wang et al., “Deep graph library: A graph-centric, highly-performant package for graph neural networks,” 2019, arXiv:1909.01315.

[22] Y. Ganin et al., “Domain-adversarial training of neural networks,” J. Mach. Learn. Res., vol. 17, no. 59, pp. 1–35, 2016.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., vol. 27, 2014, pp. 3320–3328.

[24] M. Amiri, R. Brooks, and H. Rivaz, “Fine-tuning U-net for ultrasound image segmentation: Different layers, different outcomes,” IEEE Trans. Ultrason., Ferroelectr., Freq. Control, vol. 67, no. 12, pp. 2510–2518, Dec. 2020.