Practical WiFi Indoor Localization: Unleashing the Potential of GNNs for Accuracy and Robustness

Ziqi Ye, Qiqi Xiao, Jianwei Liu, Yinghui He, *Member, IEEE*,
Guanding Yu, *Senior Member, IEEE*, and Jinsong Han, *Senior Member, IEEE*

***Abstract*—WiFi-based indoor localization, supported by comprehensive infrastructure, is considered a highly promising solution. However, practical applications of existing WiFi-based methods often struggle due to dynamic antenna configurations and potential influence from obstacles, which can undermine the reliability of channel state information and frustrate localization. Even worse, environmental changes may lead to a domain shift, further degrading localization accuracy and system robustness. To address these problems, this paper introduces GraphFi, a novel system that leverages graph neural networks (GNNs) to deliver accurate and robust localization using nearby access points (APs). GraphFi designs two types of graph structures: intra-AP graph and inter-AP graph, to maximize the use of information from all available APs. They aggregate the local features among antennas within each AP and global features across APs, effectively addressing the problem of dynamic antenna configurations. Two specialized GNNs are utilized to derive accurate user locations from these graphs. Additionally, we introduce an anomaly detection method to identify and exclude obstacle-affected APs. This method also employs a tailored GNN to mitigate influence from unpredictable obstacles. Furthermore, we integrate an unsupervised domain adaptation mechanism based on a gradient reversal layer into GNNs. This helps maintain localization performance in a cost-efficient manner and ensures sustained effectiveness in a cross-domain setting. We prototype GraphFi using commodity WiFi devices and conduct extensive experiments in various scenarios. The results demonstrate that GraphFi achieves average localization errors of 0.17 m in a single-domain setting and 0.2851 m in a cross-domain setting, surpassing existing solutions in both precision and robustness.**

***Index Terms*—WiFi, Localization, Graph Neural Networks**

I. Introduction

Accurate and robust localization has been pursued for decades, particularly in indoor environments (e.g., airports and shopping malls) where GPS signals are weak and even unavailable [1]. Towards this goal, various technologies have been explored by prior works, including WiFi [2]–[4], ultra-wideband [5], millimeter-wave [6], and ultrasonic [7]. Among these technologies, active WiFi-based localization offers a promising solution for large-scale public indoor environments due to the wide deployment of WiFi access points (APs) and the ubiquity of portable WiFi devices, such as smartphones and smartwatches. The signal sent by the WiFi transmitter arrives at the receiver after undergoing a strong line-of-sight (LoS) path and some weak non-line-of-sight paths. By extracting essential information from the LoS path, existing studies [8]–[10] have successfully utilized WiFi signals for sensing. As a subfield of WiFi sensing, research on WiFi-based indoor localization [11]–[14] has demonstrated the feasibility of using WiFi channel state information (CSI) for accurate localization in certain scenarios. Additionally, accuracy can be further enhanced by utilizing larger bandwidth [12] or incorporating CSI measurements from packets across all nearby APs [15].

Unfortunately, even with the aid of large bandwidth and multiple APs, the existing WiFi-based localization approaches, whether modeling-based [11], [15] or learning-based [16], [17], are still far from practicality since they rely on idealized assumptions, such as *"rigid" transmitter/receiver configuration* and *"clean" environment*. On the one hand, existing WiFi devices are designed primarily for communication, and their configurations of transmit and receive antennas can vary based on communication needs. Specifically, a varying subset of transmit antennas may be used for communication [18]–[20]. This variability leads to changes in the dimensionality of the CSI at the receiver, which conflicts with the fixed-dimensionality requirements of most learning-based methods [14], [21]–[23]. Moreover, in communication contexts, the indices of receive antennas may not be visible to the user and may change randomly [24]–[26]. This is problematic for modeling-based algorithms, which depend on phase differences between the received signals of adjacent antennas [27]. On the other hand, active localization depends on a stable LoS path. However, in real-world scenarios, we cannot always ensure the continuous presence of such a path between the WiFi AP and receiver all the time. Especially, in dynamic environments, such as airports and shopping malls, obstacles like passing pedestrians can obstruct the LoS path, resulting in obstacle-affected CSI and poor localization performance.

In this paper, we introduce GraphFi to address these problems by leveraging graph neural networks (GNNs) for localization. As illustrated in Figure 1, GraphFi measures the CSI by sniffing the packets transmitted by all nearby APs. Then, the collected CSI is structured into graphs and fed

Manuscript received 27 March 2025; revised 19 August 2025, 22 November 2025; accepted 01 December 2025. This work was supported in part by the National Natural Science Foundation of China under Grant No. 62372400, in part by the Postdoctoral Fellowship Program of CPSF under Grant No. GZC20241488, and in part by the Postdoctoral Research Excellence Funding Project of Zhejiang Province under Grant No. ZJ2025024. *(Corresponding authors: Yinghui He and Jianwei Liu)*

Z. Ye, Q. Xiao, Y. He, and G. Yu are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (email: {yeziqi, xiaoqiqi, 2014hyh, yuguanding}@zju.edu.cn).

J. Liu is with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China, and also with the School of Information and Electrical Engineering, Hangzhou City University, Hangzhou 310015, China (email: jianweiliu@zju.edu.cn).

J. Han is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, Zhejiang, China (e-mail: hanjinsong@zju.edu.cn).

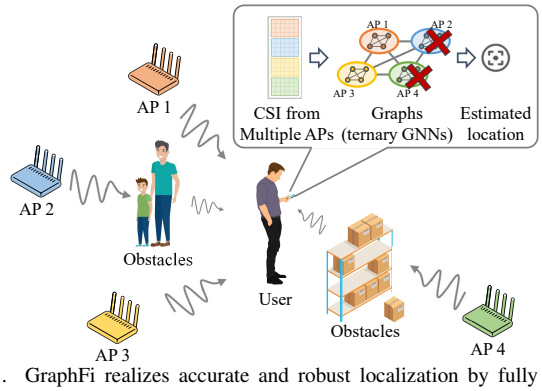


Fig. 1. GraphFi realizes accurate and robust localization by fully utilizing CSI measured from multiple APs' packets with GNNs.

into GNNs provided by the facility provider to estimate the location. Thanks to the inherent scalability and permutation invariance [28] of GNNs, GraphFi remains robust against the dynamic antenna configurations.

To implement GraphFi in practical scenarios, we address the following questions: 1) **How to make full use of the information provided by all available APs to achieve accurate localization?** To leverage GNN for localization, we need to structure CSI measurements as graphs. To this end, traditional approaches [29], [30] treat each AP as a node to construct a graph. But this would overlook the local features among antennas within each AP and simplify the features of the APs. To overcome this limitation, we propose two types of graphs: intra-AP graph and inter-AP graph. Each graph is paired with a customized GNN. The former aggregates local features among antennas within each AP, while the latter globally combines these features from all APs to provide accurate location estimates. 2) **How to avoid influence from the potential obstacles?** As aforementioned, the unpredictable obstacles like pedestrian in the environment could block the LoS path between APs and users, leading to low-quality CSI measurements and degraded localization performance. To deal with this problem, we come up with an anomaly detection method to filter out obstacle-affected CSI prior to location estimation. Specifically, we notice that the CSI with and without obstacles blocking the LoS path exhibit distinct distributions [31]. Therefore, we design an anomaly detection GNN to learn the distribution without obstacles. CSI that falls outside this learned distribution is flagged as obstacle-affected and excluded. This ensures that only the features unaffected by obstacles are used for location inference, significantly enhancing the robustness of GraphFi. 3) **How to maintain performance cost-efficiently when the environment changes?** When deploying a localization system in practice, the layout of objects in the environment may change over time. This leads to changes in the feature distributions of the CSI, which is referred to as domain shift [32], [33], ultimately resulting in a degradation of the localization system's performance. However, frequently re-collecting large amounts of CSI and annotating location labels would incur high system maintenance costs. To address this issue, we integrate a gradient reversal layer (GRL)-based unsupervised domain adaptation (UDA) mechanism into GNNs. During the process of aligning feature distributions, the GRL-based UDA helps GNNs capture domain-invariant features through adversarial training, leveraging previously collected labeled CSI (i.e., source domain data) and a small amount of re-collected unlabeled CSI (i.e., target domain data). This significantly reduces the cost of data re-collection and annotation.

We build a prototype of GraphFi with commodity WiFi devices and conduct real-world experiments to evaluate its performance. The results show that GraphFi achieves average localization errors (LEs) of 0.17 m in a single-domain setting and 0.2851 m in a cross-domain setting, maintaining high accuracy despite dynamic antenna configurations and obstacle influence. Comparisons with several existing solutions highlight GraphFi's superiority in both precision and robustness. In summary, this paper makes the following contributions:

- We propose GraphFi to achieve robust indoor WiFi localization with multiple APs, overcoming the unrealistic assumptions of a "clean" environment and a "rigid" transmitter/receiver configuration found in previous works.
- We design two types of graphs, intra-AP and inter-AP graphs, to address dynamic antenna/AP configurations and fully utilize the relative location information of APs.
- We develop an anomaly detection method to identify obstacle-affected APs and filter out low-quality CSI, significantly enhancing the robustness of localization.
- We introduce a GRL-based UDA method to maintain performance in a cost-efficient manner, ensuring the sustainability of the system in the cross-domain setting.
- We implement a prototype of GraphFi using commodity WiFi devices, and extensive experiments confirm that GraphFi can achieve accurate and robust localization in practical scenarios.

The rest of the paper is organized as follows. Section II provides a brief overview of related works. Section III introduces the background for relevant technologies and motivations for using GNNs to enhance localization robustness. Section IV provides an overview of the proposed GraphFi. Section V and Section VI detail the design of specialized graphs and corresponding ternary GNNs, respectively. Section VII outlines the implementation of GraphFi and reports its performance. The whole paper is concluded in Section VIII.

II. Related Work

This work is related to WiFi indoor localization techniques, including modeling-based and learning-based methods.

Modeling-based localization: Modeling-based methods utilize the wireless channel models and geometric relationships to estimate the location [1]. Wang *et al.* [34] propose a modeling-based localization approach that preprocesses CSI according to the target's presence in the first Fresnel zone and estimates the target location by solving power fading model equations across multiple paths. Han *et al.* [35] leverage a single WiFi AP for indoor localization by first eliminating phase and angle errors with an interpolation-based technique. They then obtain the signal propagation distance through broadband angle ranging and use triangulation to determine the target location. Zhang *et al.* [36] develop a localization system that employs a background elimination algorithm to

eliminate the static paths and extract the target reflection path using angle of arrival (AoA) and equivalent time of flight (ToF) information. Furthermore, Xie *et al*. [37] propose to jointly use four dimensions, i.e., AoA, angle of departure (AoD), ToF, and Doppler shift, to realize localization with a designed iterative algorithm. Yang *et al*. [15] introduce a joint multi-AP AoA estimation method that introduces a non-parametric AoA accuracy metric and explicitly incorporates the relationship among AoAs from different APs to enhance indoor localization robustness. To realize localization, most of the above-mentioned works rely on the CSI difference between adjacent antennas. However, in commercial WiFi devices, the indices of receive antennas may not be fixed and may change randomly, as highlighted in [24]–[26], causing these methods to become ineffective. To address this issue, we propose GraphFi that takes advantage of the permutation invariance of GNN.

Learning-based localization: As the neural networks have shown excellent performance on image-related tasks, they are also introduced to learning the complex relationship between the location and CSI for localization [38]. Intuitively, Wu *et al*. [21] treat each CSI over different antennas/subcarriers as a vector and use typical DNN to realize localization. However, the simple structure of DNN makes it hard to achieve high performance. To use complex deep networks, such as CNN, CSI can be treated as an image [39]. Wang *et al*. [14] present a CNN-based indoor localization approach that estimates stable AoA features from CSI phase differences and constructs AoA images for location prediction. In a subsequent work, the same research group introduces a ResNet-based method employing a dual-channel residual sharing architecture for indoor localization [22]. Zhu *et al*. [23] propose an indoor localization method that converts CSI phase information into images for feature extraction with a CNN and employs a broad learning model to support fast updating of localization fingerprints. However, the learning methods directly using CSI as input cannot adapt to the dynamic antenna configuration. To address those limitations, we use the GNN for WiFi localization. Different from existing works [29], [30] that simply consider APs as graph nodes, we develop two types of graphs and corresponding GNNs to solve the dynamic antenna configuration and use the location information of APs.

III. Background and Motivation

In this section, we first present the principle of WiFi localization and the practical problems we need to face. Then, we introduce the fundamentals of GNNs and GRL-based UDA. At last, we explain the key motivations for utilizing GNNs to address those problems and achieve robust localization.

A. Active WiFi Localization

The WiFi-based active localization determines the target's location by extracting information about the LoS path in the wireless channel. Thus, we start by introducing the wireless channel model. Following the typical ray-tracing model [40], [41], the wireless channel between N^{t} transmit antennas and N^{r} receive antennas can be represented by $\boldsymbol{H} = [h_{n^{\mathrm{t}},n^{\mathrm{r}}}] \in \mathbb{C}^{N^{\mathrm{t}} \times N^{\mathrm{r}}}$, where $h_{n^{\mathrm{t}},n^{\mathrm{r}}}$ can be expressed as

$$h_{n^{\mathrm{t}},n^{\mathrm{r}}} = \sum\nolimits_{l=1}^{L} \alpha_l e^{-\frac{j2\pi f^{\mathrm{c}}}{c}\left((n^{\mathrm{t}}-1)d^{\mathrm{t}}\cos(\theta_l^{\mathrm{t}})+(n^{\mathrm{r}}-1)d^{\mathrm{r}}\cos(\theta_l^{\mathrm{r}})\right)}. \quad (1)$$

In the above, α_l, θ_l^{t}, and θ_l^{r} are the path gain, AoD, and AoA for the l-th path, respectively. d^{t} and d^{r} are the transmit antenna spacing and receive antenna spacing, respectively, f^{c} is the carrier frequency, and c is the speed of light. Without loss of generality, we set $l = 1$ as the LoS path. Generally, α_1 is higher than α_l when $l \geq 2$. Existing modeling-based localization algorithms determine the receiver's location by deriving the AoA and AoD of the LoS path from Eqn. (1). Here, AoA (or AoD) is estimated using the phase difference among adjacent transmit (or receive) antennas, denoted as $d^{\mathrm{r}}\cos(\theta_l^{\mathrm{r}})$ (or $d^{\mathrm{t}}\cos(\theta_l^{\mathrm{t}})$) [27]. Unlike modeling-based methods, learning-based localization algorithms use $\boldsymbol{H}$ directly as the input to the model and establish its connection with the location by training over a collected dataset. These two types of methods can achieve high localization accuracy under some specific settings. Yet, in many real-world scenarios, we need to face the following three practical problems.

Problem 1: *Dynamic antenna configuration*. Existing WiFi systems are originally designed for efficient communication without considering the performance of localization. In practice, the antennas of the AP used for data transmission may change to adapt to the variations in the communication demands. Consequently, the number of antennas is not consistently maintained all the time, causing the dimensionality of $\boldsymbol{H}$ to fluctuate. This hinders the implementation of deep learning models as they usually require the input $\boldsymbol{H}$ to have consistent dimensionality, i.e., size. Moreover, the indices of the receive antennas at the user end can also vary depending on the communication context. Nonetheless, random variations in these indices can change the order of the row belonging to each antenna in $\boldsymbol{H}$, severely impacting modeling-based localization since accurate AoA estimation cannot be achieved.

Problem 2: *Potential obstacle influence*. In real-world scenarios, obstacles such as pedestrians can occasionally block the LoS path between the AP and the user. This can disrupt the wireless channel, deteriorating signal quality and consequently reducing localization accuracy. For example, the performance of modeling-based methods depends on the difference between α_1 and α_l, where $l \geq 2$. The obstacles can inadvertently increase α_l ($l \geq 2$) while reducing α_1, significantly diminishing the localization accuracy.

Problem 3: *Cost-efficient fine-tuning with unlabeled data*. The layout of objects in an environment often changes, e.g., adding new furniture. These changes would cause a decline in the performance of pre-trained models, particularly when the models were originally trained in a specific layout. For most existing algorithms, addressing this issue typically requires the re-collection of labeled data to fine-tune the model. However, frequent acquisition of labeled data is time-consuming and costly. In such a case, the ability to fine-tune the model using a small amount of unlabeled data would significantly reduce the operational cost in practical applications.

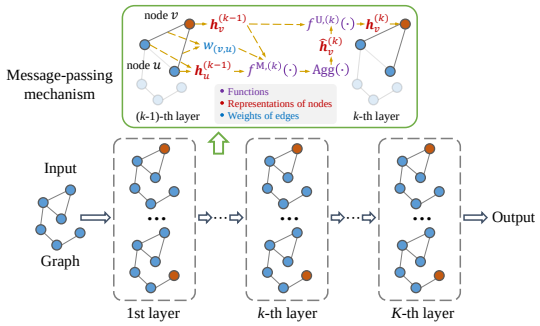


Fig. 2. The structure of GNN.

In this paper, we develop GraphFi to solve the above problems, enabling accurate localization in dynamic and obstacle-prone environments and facilitating efficient model maintenance through cost-efficient fine-tuning with unlabeled CSI.

B. GNN Basics

GNN is one class of neural networks particularly adept at processing data structured as graphs [42]. Thus, before detailing GNN, we first introduce the graph, a structure used for describing the pairwise relations between objects. A graph can be denoted as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (i.e., objects) and $\mathcal{E}$ is the set of edges. An adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ represents the connections in the graph with each element determined by the presence of edges:

$$\boldsymbol{A}[v,u] = \begin{cases} w_{(v,u)} & \text{if edge } (v,u) \in \mathcal{E} \text{ and } v \neq u, \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where $w_{(v,u)}$ is the weight (a specific type of feature) of edge (v,u). For an unweighted graph, $w_{(v,u)}$ is set as 1 for all edges. For each node $v \in \mathcal{V}$, its characteristic is represented by feature $\boldsymbol{h}_v$ and the set of its neighboring nodes is $\mathcal{N}(v)$.

GNN is built upon graph structures, as shown in Figure 2. It mainly consists of K graph convolution layers and an output layer. Different from the conventional convolution layer, the graph convolutional layer aims to update the node feature based on the information from neighboring nodes using a novel message-passing mechanism. Specifically, let $\boldsymbol{h}_v^{(k-1)}$ denote the representation of node v at the $(k-1)$-th layer. To obtain $\boldsymbol{h}_v^{(k)}$ at the k-th layer, we first calculate an intermediate feature $\hat{\boldsymbol{h}}_v^{(k)}$ for node v, as

$$\hat{\boldsymbol{h}}_v^{(k)} = \underset{u\in\mathcal{N}(v)}{\text{Agg}} \left(f^{\text{M},(k)} \left(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{h}_u^{(k-1)}, w_{(v,u)} \right) \right), \tag{3}$$

where $f^{\text{M},(k)}(\cdot)$ is the message function, e.g., a multi-layer perceptron (MLP), $\text{Agg}(\cdot)$ is the aggregation function, e.g., the summation function, and $\boldsymbol{h}_v^{(0)} = \boldsymbol{h}_v$ is the input of the GNN. Subsequently, the representation $\boldsymbol{h}_v^{(k)}$ of node v at the k-th layer is calculated with $\boldsymbol{h}_v^{(k-1)}$ and $\hat{\boldsymbol{h}}_v^{(k)}$, as

$$\boldsymbol{h}_v^{(k)} = f^{\text{U},(k)} \left(\boldsymbol{h}_v^{(k-1)}, \hat{\boldsymbol{h}}_v^{(k)} \right), \tag{4}$$

where $f^{\text{U},(k)}(\cdot)$ is the update function realized via the MLP.

Thanks to the compatibility with graph structure, GNN features two advantages: *scalability* and *permutation invariance* [28]. Scalability refers to that GNN is not limited to fixed-size input and adapts to different sizes of graphs during the training and inference phases as GNNs aggregate information from nodes and their neighboring nodes through a message-passing mechanism, regardless of the nodes' number. Permutation invariance means that the GNN is not sensitive to the order of nodes since it focuses on the connection relationship among nodes.

C. GRL-based UDA

UDA is a technique that enables the transfer of neural networks from a source domain to a target domain using unlabeled data [43]. The source domain refers to the domain where the model is pre-trained with labeled data, while the target domain is related to the source domain but has different data feature distributions, with labels typically unavailable, referring to the case where the layout of the environment varies. Among the existing UDA methods, GRL-based UDA is the most representative one, and a model that utilizes GRL-based UDA typically consists of four components: feature extractor, label predictor, GRL, and domain classifier. During transfer, the labeled data from the source domain and the unlabeled data from the target domain are input into the feature extractor for feature extraction. The extracted features are denoted as source features and target features, respectively. Then, only the source features are input into the label predictor for prediction, producing the predicted label values. Concurrently, both the source features and target features are input into the domain classifier, which attempts to distinguish whether the features come from the source domain or the target domain. The GRL, denoted as $R_\lambda(\boldsymbol{x})$, is placed between the feature extractor and the domain classifier, with the behaviors in forward and backward propagation being

$$R_\lambda(\boldsymbol{x}) = \boldsymbol{x}, \quad \frac{dR_\lambda}{d\boldsymbol{x}} = -\lambda \mathbf{I}, \tag{5}$$

respectively. During the forward propagation, the GRL performs an identity mapping on the extracted features $\boldsymbol{x}$, which means that the features are directly input to the domain classifier. During the backpropagation, the GRL reverses the gradient of the feature extractor by multiplying it by a coefficient $-\lambda$, thereby creating an adversarial training process. Thus, the domain classifier attempts to distinguish as much as possible the domain from which the features extracted by the feature extractor originate, while the feature extractor strives to extract domain-invariant features. After training, when the model is tested in the target domain, the feature extractor, which aligns the features of the source and target domains, allows the label predictor to effectively realize the task. Notably, for indoor localization tasks, the model is typically pre-trained in the source domain. Therefore, the aforementioned transfer training process is essentially a fine-tuning process, requiring only a limited number of training epochs. With its low fine-tuning overhead and reduced costs of data re-collection and annotation, GRL-based UDA provides a cost-efficient maintenance solution for indoor localization.

D. How Our Solutions Benefit from GNN?

The core challenge of Problem 1 stems from the varying size of $\boldsymbol{H}$ and the random order of its rows. Fortunately,

the scalability and permutation invariance of GNN provide a solution to this problem. Previous approaches [29], [30] have shown that treating each AP as a node to construct the graph is feasible, but this does not fully leverage the scalability of GNNs. A better approach is to consider each transmitter-receiver (TX-RX) antenna pair as a node. However, this straightforward graph structure overlooks critical information in a multi-AP scenario, specifically the connections among APs, which can negatively affect the localization performance. To address this issue, we propose two novel graph structures for the antenna-level and AP-level respectively, namely intra-AP graph and inter-AP graph. The AP level captures the connections among APs, while the antenna level represents the connections between the transmit antennas of a single AP and the receive antennas of the user. This two-tiered approach enables GraphFi to fully leverage the CSI from all APs and the capabilities of GNNs, resulting in highly accurate localization.

Our approach for addressing Problem 2, where potential obstacles may affect CSI, draws inspiration from ancient mariners. Ancient mariners navigated by using stars in the night sky while sailing at sea. When clouds obscured some stars, they would depend on other visible stars to navigate. Similarly, indoor localization tasks bear resemblance to the navigation challenges faced by mariners. In large indoor environments, numerous APs are densely deployed to maintain communication performance. These APs can be likened to stars that can be utilized for localization purposes. However, obstacles within the environment may attenuate signals from certain APs. Incorporating the CSI from these APs for localization can potentially introduce noise into the data distribution, leading to degraded localization performance. To address this issue, akin to ancient mariners, we propose an anomaly detection method on the CSI between the user equipment (UE) and various APs. This method assesses the impact of obstacles on the CSI of each AP and selectively uses only the CSI that is unaffected by obstacles for localization. Although this would change the dimensionality of $\boldsymbol{H}$ and pose a challenge to traditional deep neural networks, GraphFi leverages the exceptional scalability of GNNs to maintain effective performance in such scenarios.

To address Problem 3, we integrate GRL-based UDA into our GNN framework. This technique offers several advantages that are particularly beneficial for indoor localization tasks. Firstly, it enables efficient optimization through standard backpropagation, avoiding the complex and often unstable adversarial training required by GAN-based methods. Secondly, GRL-based UDA enhances robustness by learning domain-invariant features without relying on target domain labels, which is essential for adapting to dynamic indoor environments where labeled data is scarce or unavailable. Additionally, this approach can seamlessly integrate with existing deep learning frameworks. By leveraging these advantages, the integration of GRL-based UDA with our GNN framework allows GraphFi to adapt to changes in the indoor environment without the need for labeled target data, thereby facilitating cost-efficient system maintenance.

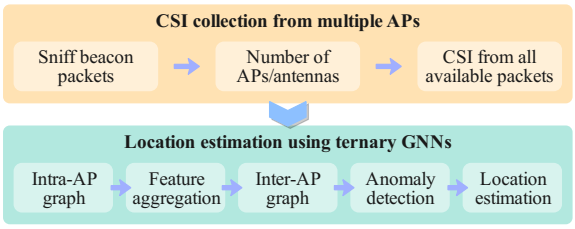


Fig. 3. Overview of GraphFi.

IV. GraphFi Overview

Towards accurate and robust localization using CSI extracted from multiple APs' packets, GraphFi follows the workflow outlined in Figure 3. The process consists of two phases: CSI collection from multiple APs and location estimation using ternary GNNs. To localize a user in an indoor environment, the process begins with sniffing the beacon packets to identify the number of nearby APs and further identify the number of transmit antennas for each AP from the high-throughput capabilities information contained in the beacon packets [44]. Next, the user measures the CSI between the nearby APs and itself by capturing all available packets transmitted by the APs. Using the collected CSI, the user constructs an intra-AP graph (introduced in Section V-A). This graph is then fed into a feature aggregation GNN to obtain the characteristic representation (CR) of each AP and construct an inter-AP graph (introduced in Section V-B). Subsequently, GraphFi uses an anomaly detection GNN to filter out the CRs of obstacle-affected APs. Finally, the inter-AP graph, after removing obstacle-affected APs, is input into a location estimation GNN to determine the user's location. The intra-AP and inter-AP graphs will be introduced in Section V. The feature aggregation, anomaly detection, and location estimation GNNs will be detailed in Section VI. Additionally, the training strategy of GraphFi in a single-domain setting and the fine-tuning strategy in a cross-domain setting will also be described in Section VI.

V. Graph Construction

To fully leverage all available APs for accurate localization, we design two specialized graphs: the intra-AP graph and inter-AP graph. These graphs are crafted to effectively integrate and structure the CSI provided by the APs. In the following, we first outline the variables needed to construct these graphs and then give the construction details.

To be aware of nearby available APs, GraphFi scans for the beacon packets in the environment and identifies the source of each packet based on its media access control address. The total number of the available APs can be denoted as I. Additionally, the number of transmit antennas for each AP, which corresponds to the number of supported data streams indicated in the beacon packet, is denoted by N^{t}.[1] Then, we can structure the graph based on the above information. Recall that our goal is to make full use of the knowledge provided by APs, we design the intra-AP graph and inter-AP

[1]For ease of understanding, we assume that all APs are equipped with the same number of antennas in this paper. However, the proposed design can be easily extended to accommodate general scenarios where the number of antennas varies among different APs.

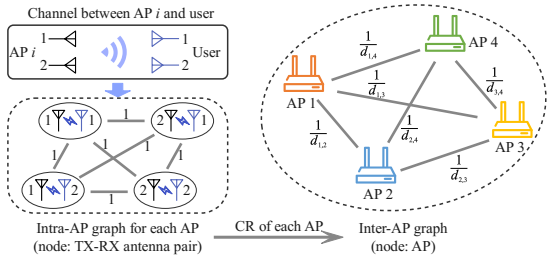


Fig. 4. The designed two types of graphs.

graph to exploit local information among TX-RX antennas of each AP and global information among APs, respectively. The relationship between them is illustrated in Figure 4. First, GraphFi constructs an intra-AP graph for each AP, in which each TX-RX antenna pair is regarded as a node to offer a fine-grained utilization of the local CSI information. Then, each AP is considered as a node to construct an inter-AP graph. It globally combines the local information from all the APs to estimate the user location.

A. Intra-AP Graph

GraphFi assigns each AP an intra-AP graph to collect local information among antennas, where each TX-RX antenna pair corresponds to a node. Formally, the i-th intra-AP graph can be described by:

$$G^{(i)} = (\mathcal{V}^{(i)}, \mathcal{E}^{(i)}), \tag{6}$$

where $\mathcal{V}^{(i)}$ is the set of TX-RX antenna pairs and $\mathcal{E}^{(i)}$ is the set of corresponding edges. Particularly, the number of nodes (i.e., antenna pairs) has an upper limit $N^{\mathrm{t}}N^{\mathrm{r}}$. The actual number of the nodes may be less than $N^{\mathrm{t}}N^{\mathrm{r}}$ and vary with time, as not all transmit antennas are necessarily used for communication in practice. As for the edge, any two TX-RX antenna pairs are considered connected since they are highly correlated. Consequently, the adjacency matrix $\boldsymbol{A}^{(i)}$ has all non-diagonal elements set to 1 and all diagonal elements set to 0.

B. Inter-AP Graph

To construct an inter-AP graph for combining the local information of all available APs, each AP contributes as a node, with the number of nodes being I. Formally, the inter-AP graph can be described as:

$$G^{\mathrm{AP}} = (\mathcal{V}^{\mathrm{AP}}, \mathcal{E}^{\mathrm{AP}}), \tag{7}$$

where $\mathcal{V}^{\mathrm{AP}}$ is the set of APs and $\mathcal{E}^{\mathrm{AP}}$ is the set of edges among APs. We assume that any two APs are also connected, given that they are all in close proximity to the user. To better reflect real-world relationships among APs, we incorporate the distance between each pair of APs into the adjacency matrix $\boldsymbol{A}^{\mathrm{AP}} \in \mathbb{R}^{I\times I}$. Let d_{i_1,i_2} denote the distance between AP i_1 and AP i_2, we have

$$\boldsymbol{A}^{\mathrm{AP}}[i_1,i_2] = \begin{cases} 1/d_{i_1,i_2} & \text{if } i_1 \neq i_2, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

$\boldsymbol{A}^{\mathrm{AP}}[i_1,i_2]$ reflects the correlation between the CSI from AP i_1 and that from AP i_2, where a smaller distance d_{i_1,i_2} indicates a stronger correlation.

C. Superiority of Our Graphs

Compared to the GNN construction methods proposed by existing localization literature [29], [30], our customized graph construction method bears the following advantages: (1) **Mitigation of data disorder issue:** our method effectively addresses problems associated with data disorder within the CSI matrix caused by random indices of antennas. While random indices lead to sequential exchanges of nodes within intra-AP graphs, the permutation invariance ensures that the connection relationships and overall structure of the two graphs remain unaffected, that is, these changes do not impact the GNN output. (2) **Fine-grained information utilization:** unlike existing methods [29], [30] that only treat one AP as a node, our approach adopts a more detailed perspective. We first represent the CSI between a TX-RX antenna pair as node features in the intra-AP graph. These node features are aggregated to form the CR of AP. Subsequently, the CRs are combined with AP distance information through the inter-AP graph to determine the UE location. Such a countermeasure not only utilizes local CSI data but also makes full use of the global information among APs, thereby enhancing the overall performance of the localization system.

VI. Localization via Ternary GNNs

As mentioned above, we design two types of graphs to achieve fine-grained information utilization. In this section, we respectively develop two GNNs to leverage the benefits of the two graphs for precise localization. Additionally, to mitigate potential influence from low-quality CSI of some obstructed APs, we also design another GNN to perform anomaly detection. As a result, we employ three distinct GNNs, collectively referred to as ternary GNNs (as shown in Figure 5) to achieve local feature aggregation in intra-AP graphs, anomaly detection for the CRs, and location estimation with the global inter-AP graph, respectively. In the following, we will first introduce how the ternary GNNs accomplish the localization tasks during the inference phase, and then present the training and fine-tuning strategy of the ternary GNNs.

A. Feature Aggregation GNN

The purpose of this GNN is to aggregate the local features of TX-RX antenna pairs within each intra-AP graph. Its structure is illustrated in Figure 6. We use the CSI of each TX-RX antenna pair as the node features, denoted as $\boldsymbol{h}_n^{(i)} \in \mathbb{C}^{N^{\mathrm{f}}\times 1}$, where N^{f} is the number of subcarriers and $n \in \{1,\cdots,N^{\mathrm{t}}N^{\mathrm{r}}\}$. Three graph convolutional layers are employed to uncover potential relationships among nodes through the message-passing mechanism. Specifically, in the k-th graph convolutional layer, given the input $\boldsymbol{h}_n^{(i,k-1)}$, the output after the message-passing can be expressed as:

$$\boldsymbol{h}_n^{(i,k)} = \boldsymbol{W}_1^{(k)}\boldsymbol{h}_n^{(i,k-1)} + \boldsymbol{W}_2^{(k)} \sum_{n'\in\mathcal{N}(n)} \boldsymbol{A}^{(i)}[n,n']\boldsymbol{h}_{n'}^{(i,k-1)}, \tag{9}$$

where $\boldsymbol{W}_1^{(k)}$ and $\boldsymbol{W}_2^{(k)}$ are the trainable parameters. Moreover, each graph convolutional layer is followed by a batch normalization (BN) function [45] and a rectified linear unit

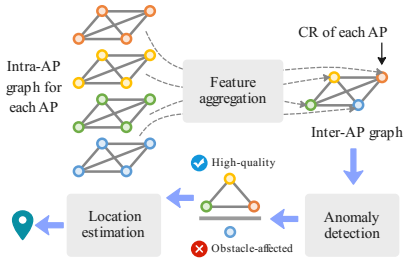


Fig. 5. Localization using ternary GNNs.

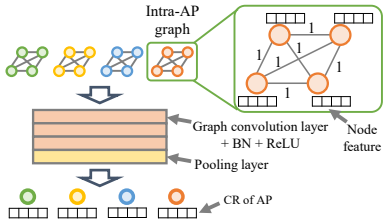


Fig. 6. Feature aggregation GNN.

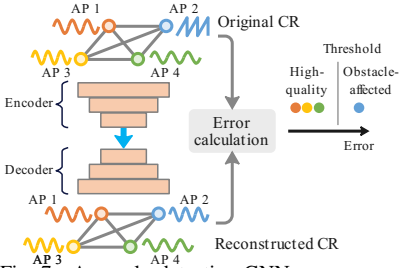


Fig. 7. Anomaly detection GNN.

(ReLU) activation function [46]. BN helps mitigate biases in the data distribution, enhancing the stability and speed of the training process. ReLU introduces nonlinear transformations, improving the network's learning capability. After the processing of the ReLU activation, the output $\boldsymbol{h}_n^{(i,k)}$ is obtained and serves as the input of the next convolutional layer.

After three convolutional layers, we use a global average pooling layer (denoted as $\mathcal{P}(\cdot)$) to aggregate the features of all TX-RX antenna pairs into a single representation for this AP, i.e., CR. The CR for the i-th AP can be expressed as:

$$\boldsymbol{g}^{(i)} = \mathcal{P}\left(\{\boldsymbol{h}_n^{(i,3)} | n = 1, \cdots, N^{\mathrm{t}} N^{\mathrm{r}}\}\right). \tag{10}$$

Since each node is fully connected with all edges set to 1, the order of the nodes, i.e., the indices of antennas, does not affect the CR of each AP. Additionally, the number of nodes, which corresponds to the number of TX-RX antenna pairs, also does not impact the aggregation process. This is because we use a global average pooling function to ensure that the size of the aggregated features remains consistent when the number of nodes changes.

B. Anomaly Detection GNN

After the feature aggregation in each intra-AP graph, we obtain the CR of each AP, and it can be used for subsequent localization. However, as aforementioned, due to that potential obstacle may impede the LoS path between the AP and the user, the CRs of some APs may be obstacle-affected. To address this issue, we need to detect the obstacle-affected AP and only use the CRs of unobstructed APs for localization. Thus, we develop an anomaly detection GNN to examine the quality of each CR.[2]

Given that influence from potential obstacles is unpredictable, collecting CSI samples under all possible scenarios is challenging. Instead, it is more practical to collect CSI samples in the absence of obstacles. To achieve our objective of detecting an obstacle-affected AP, we need to differentiate between CSI with and without obstacles. According to the channel model [40], a channel without obstacle blockage follows a Rice distribution, while a channel with obstacles follows a Rayleigh distribution. Therefore, we can analyze the distribution of CSI samples collected without obstacles and use this information to facilitate anomaly detection.

To achieve this, we employ an autoencoder [47] to learn the distribution of CSI that is unaffected by obstacles. As illustrated in Figure 7, the input of the autoencoder is the inter-AP graph with the CR of each AP being $\boldsymbol{g}^{(i)}$. The autoencoder contains two main components: encoder and decoder. The encoder utilizes three graph convolutional layers to compress the CRs of all APs from high-dimensional vectors to low-dimensional representations. The operation of the k-th graph convolutional layer in the encoder can be represented as:

$$\boldsymbol{g}^{(i,k)} = \boldsymbol{W}_1^{\mathrm{A},(k)} \boldsymbol{g}^{(i,k-1)} + \boldsymbol{W}_2^{\mathrm{A},(k)} \sum_{i' \in \mathcal{N}(i)} \boldsymbol{A}^{\mathrm{AP}}[i, i'] \boldsymbol{g}^{(i,k-1)}, \tag{11}$$

where $\boldsymbol{W}_1^{\mathrm{A},(k)}$ and $\boldsymbol{W}_2^{\mathrm{A},(k)}$ are the trainable parameters, and $\boldsymbol{g}^{(i,0)} = \boldsymbol{g}^{(i)}$ is the input of the autoencoder. Each graph convolutional layer is followed by a ReLU activation function. The decoder has a similar three-layer structure as the encoder, and the operation of each graph convolutional layer in the decoder is analogous to that in the encoder. The key difference is that the decoder reconstructs the encoded CRs, i.e., transforms them from low-dimensional vectors back into high-dimensional CRs.

As mentioned above, the autoencoder only needs to "see" the distribution of clean CSI without obstacle influence. To identify the CRs of obstacle-affected APs, we define the reconstruction error as:

$$\epsilon^{(i)} = ||\hat{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i)}||_2, \tag{12}$$

where $\hat{\boldsymbol{g}}^{(i)}$ denotes the reconstructed CR for the i-th AP. We then set an empirical threshold η for the reconstruction error, determined from experimental observations as the 95 % quantile of the errors under unobstructed APs. Since the distribution of obstacle-affected CSI differs from that of unaffected CSI, the reconstruction error for obstacle-affected CRs will be significantly high. During the inference phase, only CRs with a reconstruction error less than η are used to estimate location.

C. Location Estimation GNN

The location estimation GNN is tasked with globally integrating the CRs from all unobstructed APs (i.e., taking inter-AP graph as input) to estimate the user's location. First, the inter-AP graphs are fed into the inter-AP graph feature extractor, which consists of two graph convolutional layers and a global average pooling layer. Specifically, the two graph convolutional layers employ a message-passing mechanism to uncover potential relationships among APs. The operation of each graph convolutional layer is described in Eqn. (11). Each graph convolutional layer is followed by a BN function and a ReLU activation function. After the final ReLU activation, a global average pooling layer is applied to average the

[2]Here, "anomaly" refers to features that are uninformative for localization rather than abnormalities in communication.

convolutional results of all CRs to obtain the extracted global features. Ultimately, the label predictor composed of a fully-connected (FC) layer is used to map the extracted global features to the user's location.

At this point, we have completed the introduction of the inference process of the ternary GNNs. It consists of three modules: the feature aggregation GNN, the anomaly detection GNN, and the location estimation GNN. Let I denote the number of APs, and N^{t} and N^{r} the numbers of transmit and receive antennas, respectively. Treating the input/output feature dimensions of each layer as constants and ignoring lower-order terms, the computational complexities of the three modules are as follows: feature aggregation GNN: $\mathcal{O}\big(I(N^{\mathrm{t}}N^{\mathrm{r}})^2\big)$, anomaly detection GNN: $\mathcal{O}\big(I^2\big)$, and location estimation GNN: $\mathcal{O}\big(I^2\big)$. Therefore, the overall computational complexity of GraphFi's ternary GNNs during inference is $\mathcal{O}\big(I(N^{\mathrm{t}}N^{\mathrm{r}})^2 + I^2\big)$. The rest of this section will describe the training strategy in the single-domain setting (i.e., source domain) and the fine-tuning strategy in the cross-domain setting.

D. Training and Fine-Tuning Strategy

To make the aforementioned ternary GNNs possess their presumptive prediction abilities, we need to construct a dataset and train over it based on tailored optimization objectives. Particularly, in the single-domain setting, ternary GNNs are trained using labeled data from the source domain, and the training process can be divided into two stages. In the first stage, the feature aggregation and location estimation GNNs are jointly optimized. In the second stage, the anomaly detection GNN is optimized based on the well-trained feature aggregation GNN. In the cross-domain setting, labeled data previously collected from the source domain and a small amount of re-collected unlabeled data from the target domain are available for fine-tuning. Similar to the training process, the fine-tuning process can also be divided into two stages. The difference is that, in the first stage, both labeled data from the source domain and unlabeled data from the target domain are used to fine-tune the feature aggregation GNN and location estimation GNN. In the second stage, the anomaly detection GNN is fine-tuned solely based on the unlabeled data. In the following, we will first detail our data construction method, then introduce the components of the ternary GNNs that are activated only during the training or fine-tuning phases, and finally provide the training and fine-tuning details.

1) Dataset Construction: A piece of training or fine-tuning data is composed of two parts: the intra-AP graphs of all APs and the corresponding adjacency matrix of the inter-AP graph. To collect them, we let a user move in the target environment to capture the packets of all available APs, assuming that all transmit antennas of each AP are used for communication. Then, we construct intra-AP graphs and build adjacency matrices according to the methods introduced in Section V-A. Each piece of training data from the source domain is then annotated with the actual location of the user, while the fine-tuning data from the target domain lacks location labels.

2) Component Activation: We now introduce the components of the ternary GNNs that are activated only during the training or fine-tuning phases, which are disabled during the inference phase introduced in Section VI-A–Section VI-C. For the feature aggregation GNN, although the training conditions are typically ideal, some transmit antennas of an AP may be inactive during communication in practice. To address this issue, before feeding the intra-AP graphs into the graph convolutional layers, the feature aggregation GNN first performs data augmentation by simulating such scenarios, that is, randomly dropping some nodes within a portion of intra-AP graphs. This augmentation can enhance the generality of the dataset regarding dynamic antenna configuration. For the anomaly detection GNN, each graph convolutional layer is followed by a ReLU activation function and a dropout layer [48], with the exception of the final graph convolutional layer. The dropout layer helps mitigate overfitting. For the location estimation GNN, during the inference phase, the anomaly detection GNN will filter out the CRs from obstacle-affected APs, which may result in a change in the number of nodes in the inter-AP graphs. Similar to the feature aggregation GNN, the location estimation GNN also performs data augmentation by simulating such cases, that is, randomly dropping some nodes within a portion of the inter-AP graphs. The components mentioned above operate in both the training and fine-tuning phases. Additionally, the UDA module placed after the inter-AP graph feature extractor is activated only during the fine-tuning phase for the cross-domain setting. As shown in Figure 8, the UDA module consists of the GRL and the domain classifier, and the domain classifier comprises two FC layers with a ReLU activation function between them.

3) Two-Stage Training: Now, we can proceed to train ternary GNNs using the constructed dataset. In the single-domain setting, all data are from the source domain and annotated with location labels. We adopt a two-stage training strategy in this setting, as shown in Algorithm 1. In the first training stage, we jointly train the feature aggregation GNN and location estimation GNN. Since there is no domain shift in the single-domain setting, the UDA module of the location estimation GNN is disabled. The loss function in the first training stage can be expressed as:

$$\mathcal{L}^{\mathrm{L}} = ||\hat{\boldsymbol{y}}^{\mathrm{L}} - \boldsymbol{y}^{\mathrm{L}}||_2, \tag{13}$$

where $\hat{\boldsymbol{y}}^{\mathrm{L}}$ is the estimated location and $\boldsymbol{y}^{\mathrm{L}}$ is the actual location, both of which are two-dimensional planar coordinates. By minimizing $\mathcal{L}^{\mathrm{L}}$, the parameters of both feature aggregation GNN and location estimation GNN are updated. After training, we can use these two GNNs to predict the UE locations.

Nevertheless, GraphFi still encounters challenges when dealing with data contaminated by obstacle-affected APs. To address this issue, we introduce a second training stage to optimize the anomaly detection GNN. Specifically, we also perform joint optimization by training both the feature aggregation GNN and the anomaly detection GNN together. However, during this stage, only the parameters of the anomaly detection GNN are updated, while the parameters of the feature aggregation GNN remain fixed. The optimization objective of the anomaly detection GNN is to minimize the difference (i.e., reconstruction error) between the input CR and the reconstructed CR, enabling its autoencoder to learn

Algorithm 1: Two-Stage Training Strategy.

Input: Source domain dataset $\mathcal{D}_{\mathrm{S}} = \{(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}, \boldsymbol{A}^{\mathrm{AP}}, \boldsymbol{y}^{\mathrm{L}})\}$, number of epochs P_{T1}, P_{T2}, and learning rate η_{T1}, η_{T2};
Output: Trained parameters Θ^{F}, Θ^{A}, and Θ^{L};
Stage 1: Joint training of feature aggregation GNN ($\mathrm{GNN}^{\mathrm{F}}$) & location estimation GNN ($\mathrm{GNN}^{\mathrm{L}}$).

Initialize: Randomly initialize $\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}$;
for *epoch* $p = 1$ **to** P_{T1} **do**
  foreach *minibatch* $\mathcal{B} \subset \mathcal{D}_{\mathrm{S}}$ **do**
    $\boldsymbol{g}^{(i)} = \mathrm{GNN}^{\mathrm{F}}\big(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}\big)$;
    $\hat{\boldsymbol{y}}^{\mathrm{L}} = \mathrm{GNN}^{\mathrm{L}}\big(\boldsymbol{g}^{(i)}, \boldsymbol{A}^{\mathrm{AP}}\big)$;
    $\mathcal{L}^{\mathrm{L}} = ||\hat{\boldsymbol{y}}^{\mathrm{L}} - \boldsymbol{y}^{\mathrm{L}}||_2$;
    $\{\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}\} \leftarrow \{\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}\} - \eta_{\mathrm{T1}} \nabla_{\{\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}\}} \mathcal{L}^{\mathrm{L}}$;
  end
end

Stage 2: Training of anomaly detection GNN ($\mathrm{GNN}^{\mathrm{A}}$).

Initialize: Randomly initialize Θ^{A}, and freeze Θ^{F};
for *epoch* $p = 1$ **to** P_{T2} **do**
  foreach *minibatch* $\mathcal{B} \subset \mathcal{D}_{\mathrm{S}}$ **do**
    $\boldsymbol{g}^{(i)} = \mathrm{GNN}^{\mathrm{F}}\big(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}\big)$;
    $\hat{\boldsymbol{g}}^{(i)} = \mathrm{GNN}^{\mathrm{A}}(\boldsymbol{g}^{(i)})$;
    $\mathcal{L}^{\mathrm{A}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} ||\hat{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i)}||_2$;
    $\Theta^{\mathrm{A}} \leftarrow \Theta^{\mathrm{A}} - \eta_{\mathrm{T2}} \nabla_{\Theta^{\mathrm{A}}} \mathcal{L}^{\mathrm{A}}$;
  end
end

Algorithm 2: Two-Stage Fine-Tuning Strategy.

Input: Source domain dataset $\mathcal{D}_{\mathrm{S}} = \{(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}, \boldsymbol{A}^{\mathrm{AP}}, \boldsymbol{y}^{\mathrm{L}}, y^{\mathrm{D}})\}$, target domain dataset $\mathcal{D}_{\mathrm{T}} = \{(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}, \boldsymbol{A}^{\mathrm{AP}}, y^{\mathrm{D}})\}$, number of epochs P_{F1}, P_{F2} and learning rate η_{F1}, η_{F2};
Output: Fine-tuned parameters Θ^{F}, Θ^{A}, and Θ^{L};
Stage 1: Joint fine-tuning of feature aggregation GNN ($\mathrm{GNN}^{\mathrm{F}}$) & location estimation GNN ($\mathrm{GNN}^{\mathrm{L}}$).

Initialize: Load the trained $\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}$;
# The UDA module is activated in this stage.
for *epoch* $p = 1$ **to** P_{F1} **do**
  $\lambda = \frac{2}{1+\exp(-10\,p)} - 1$;
  foreach *minibatch* $\mathcal{B}_{\mathrm{S}} \subset \mathcal{D}_{\mathrm{S}}, \mathcal{B}_{\mathrm{T}} \subset \mathcal{D}_{\mathrm{T}}$ **do**
    $\boldsymbol{g}^{(i)} = \mathrm{GNN}^{\mathrm{F}}\big(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}\big)$;
    $\hat{\boldsymbol{y}}^{\mathrm{L}}, \hat{y}^{\mathrm{D}} = \mathrm{GNN}^{\mathrm{L}}\big(\boldsymbol{g}^{(i)}, \boldsymbol{A}^{\mathrm{AP}}, \lambda\big)$;
    $\mathcal{L}^{\mathrm{L}} = ||\hat{\boldsymbol{y}}^{\mathrm{L}} - \boldsymbol{y}^{\mathrm{L}}||_2$;
    $\mathcal{L}^{\mathrm{D}} = -[y^{\mathrm{D}} \cdot \log(\hat{y}^{\mathrm{D}}) + (1 - y^{\mathrm{D}}) \cdot \log(1 - \hat{y}^{\mathrm{D}})]$;
    $\mathcal{L} = \mathcal{L}^{\mathrm{L}} + \mathcal{L}^{\mathrm{D}}$;
    $\{\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}\} \leftarrow \{\Theta^{\mathrm{F}}, \Theta^{\mathrm{L}}\} - \eta_{\mathrm{F1}} \nabla \mathcal{L}$;
    # Here, Θ^{L} consists of the learnable parameters of the UDA module.
  end
end

Stage 2: Fine-tuning of anomaly detection GNN ($\mathrm{GNN}^{\mathrm{A}}$).

Initialize: Load the trained Θ^{A}, and freeze Θ^{F};
for *epoch* $p = 1$ **to** P_{F2} **do**
  foreach *minibatch* $\mathcal{B}_{\mathrm{T}} \subset \mathcal{D}_{\mathrm{T}}$ **do**
    $\boldsymbol{g}^{(i)} = \mathrm{GNN}^{\mathrm{F}}\big(\boldsymbol{h}_n^{(i)}, \boldsymbol{A}^{(i)}\big)$;
    $\hat{\boldsymbol{g}}^{(i)} = \mathrm{GNN}^{\mathrm{A}}(\boldsymbol{g}^{(i)})$;
    $\mathcal{L}^{\mathrm{A}} = \frac{1}{|\mathcal{B}_{\mathrm{T}}|} \sum_{i \in \mathcal{B}_{\mathrm{T}}} ||\hat{\boldsymbol{g}}^{(i)} - \boldsymbol{g}^{(i)}||_2$;
    $\Theta^{\mathrm{A}} \leftarrow \Theta^{\mathrm{A}} - \eta_{\mathrm{F2}} \nabla_{\Theta^{\mathrm{A}}} \mathcal{L}^{\mathrm{A}}$;
  end
end

the distribution of the channel without obstacle influence. The loss function $\mathcal{L}^{\mathrm{A}}$ for this training stage can be expressed as:

$$\mathcal{L}^{\mathrm{A}} = \sum\nolimits_{i=1}^{I} \epsilon^{(i)} / I. \tag{14}$$

After the two-stage training, the ternary GNNs can be employed for precise and robust localization. First, the intra-AP graphs from all available APs are fed into the feature aggregation GNN to generate the CRs for each AP. Next, the anomaly detection GNN is used to identify any low-quality CRs with the inter-AP graph. Only the CRs from unobstructed APs, along with their corresponding adjacency matrix, are then input into the location estimation GNN for localization.

4) Two-Stage Fine-Tuning: In the cross-domain setting, both source and target domain data are assigned domain labels to indicate whether they originate from the source or target domain, but only source domain data have location labels. Under this setting, the UDA module of the location estimation GNN is activated. Similar to the single-domain setting, we adopt a two-stage fine-tuning strategy in the cross-domain setting, as shown in Algorithm 2. In the first fine-tuning stage, the feature aggregation GNN and the location estimation GNN are still fine-tuned jointly. However, the fine-tuning process differs from the training process in the single-domain setting. It is important to note that in the first fine-tuning stage, the feature aggregation GNN and the inter-AP graph feature extractor of the location estimation GNN act as the feature extractor described in Section III-C. Specifically, intra-AP graphs from both domains are fed into the feature aggregation GNN, and the resulting inter-AP graphs are then input into the inter-AP graph feature extractor of the location estimation GNN. Then, the extracted features are obtained and referred to as source features and target features, respectively. Subsequently, as illustrated in Figure 8, only the source features are input into the label predictor to obtain the estimated location, denoted as $\hat{\boldsymbol{y}}^{\mathrm{L}}$, similar to the procedure in the training process. Concurrently, both source and target features are input into the UDA module to derive the estimated domain, $\hat{y}^{\mathrm{D}}$, which is specific to the fine-tuning process. The loss function in the first fine-tuning stage can be expressed as:

$$\mathcal{L} = \mathcal{L}^{\mathrm{L}} + \mathcal{L}^{\mathrm{D}}, \tag{15}$$

where $\mathcal{L}^{\mathrm{L}}$ and $\mathcal{L}^{\mathrm{D}}$ are the location loss and domain classification loss, respectively. $\mathcal{L}^{\mathrm{L}}$ is defined by Eqn. (13), and $\mathcal{L}^{\mathrm{D}}$ is defined as:

$$\mathcal{L}^{\mathrm{D}} = -[y^{\mathrm{D}} \cdot \log(\hat{y}^{\mathrm{D}}) + (1 - y^{\mathrm{D}}) \cdot \log(1 - \hat{y}^{\mathrm{D}})], \tag{16}$$

where $y^{\mathrm{D}} \in \{0, 1\}$ is the domain label (0: source domain, 1: target domain) and $\hat{y}^{\mathrm{D}}$ denotes the predicted value of the

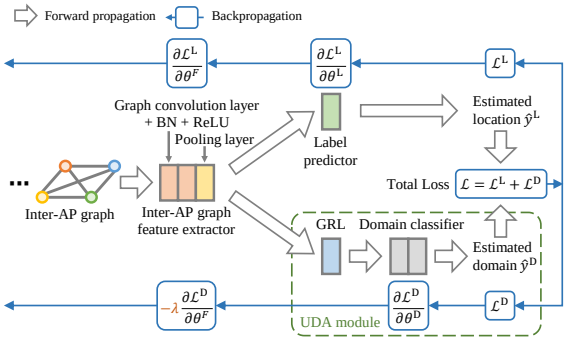


Fig. 8. This figure illustrates how UDA is integrated with the proposed ternary GNNs, and it is only activated during the fine-tuning stage.

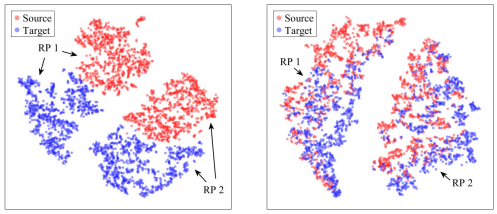
domain. When minimizing $\mathcal{L}$, the backpropagation is shown in Figure 8, where θ^{L}, θ^{D}, and θ^{F} are learnable parameters of the label predictor, the domain classifier, and the feature extractor. It can be observed that during backpropagation, the gradient is multiplied by a coefficient of $-\lambda$ when it passes through the GRL, thereby creating an adversarial training process between the feature extractor and the domain classifier. λ is a dynamic weighting parameter of GRL, which is defined as:

$$\lambda = 2/(1 + \exp(-10p)) - 1, \tag{17}$$

where p denotes the index of the current epoch in the fine-tuning stage. The gradual increase of λ from 0 to 1 enables the feature extractor (i.e., the feature aggregation GNN and the inter-AP graph feature extractor of the location estimation GNN) to capture domain-invariant features.

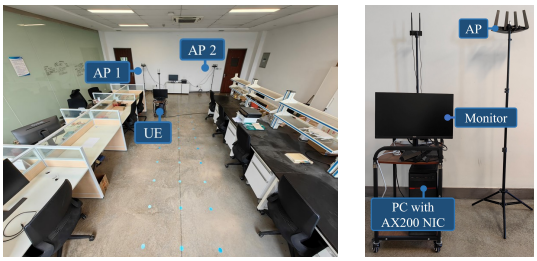
In the second fine-tuning stage, similarly, the feature aggregation GNN and the anomaly detection GNN are fine-tuned jointly. The parameters of the anomaly detection GNN are updated, while the parameters of the feature aggregation GNN remain fixed. Since this stage is unsupervised, it does not require location labels. Consequently, only the intra-AP graphs from the target domain are input into the feature aggregation GNN, enabling the anomaly detection GNN to capture the distribution of the normal CSI in the target domain. Here, the normal CSI refers to the CSI obtained under stable conditions without dynamic antenna configuration or influence from potential obstacles. Overall, the fine-tuning process in the second stage for the cross-domain setting is analogous to that of the single-domain setting.

After the two-stage fine-tuning, the ternary GNNs are capable of achieving precise and robust localization in the target domain. In the first fine-tuning stage, intra-AP graphs from both source and target domains are input into the feature aggregation GNN, where the UDA module of the location estimation GNN facilitates the extraction of domain-invariant features through adversarial training. In the second stage, only the intra-AP graphs from the target domain are input into the feature aggregation GNN, allowing the anomaly detection GNN to capture the distribution of the normal CSI in the target domain. This methodology enables GraphFi to be fine-tuned using a small amount of unlabeled target domain data, thereby effectively addressing problems related to dynamic antenna configurations and potential obstacle influence within the target domain in a cost-efficient manner. To show the effectiveness, we plot the t-SNE for two reference points (RPs) with the output features from the global average pooling layer



(a) Without UDA. (b) With UDA.

Fig. 9. t-SNE visualization of GraphFi.



(a) Experiment environment. (b) Prototype.

Fig. 10. GraphFi implementation: (a) experiment layout and (b) prototype.

of the location estimation GNN in Figure 9. One can see that the extracted features without UDA differ significantly between the source and target domains for the same RP. In contrast, after applying UDA, the points corresponding to the same RP in different domains are clustered together, indicating that the model successfully learns domain-invariant features.

VII. Evaluation

This section first describes the implementation of GraphFi, and then details its localization performance.

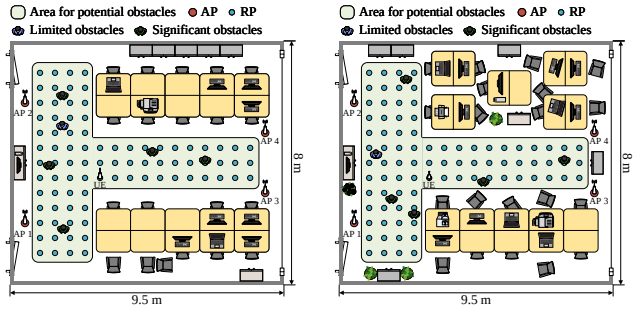
A. Implementation

*1) **Experiment Setup**:* We conduct experiments in an 8 m×9.5 m office, simulating a public indoor environment, as illustrated in Figure 10. The office contains four APs, specifically TP-LINK AX5400 routers, positioned at four distinct locations. A personal computer equipped with a two-antenna AX200 network interface card serves as the UE, and the antenna spacing is 6 cm. The antennas for APs are mounted at different heights: 1.60 m for AP 1 and AP 2, and 1.75 m for AP 3 and AP 4, representing various locations and elevations. During the collection process, four APs work in the 802.11ax WiFi standard with the carrier frequency being 5.25 GHz and bandwidth being 80 MHz. The user utilizes PicoScenes [49] tool to sniff the communication packets transmitted by the APs and measure CSI between the APs and itself, with the CSI sampling rate being 100 Hz. 85 RPs are selected as the user's possible locations, with a distance of 0.5 m between two adjacent RPs.

*2) **Data Collection**:* Data collection is conducted in both the source and target domains. The setup mentioned above is used for both source and target domain data collection. As shown in Figure 11(a), we begin by collecting data from the source domain, where the height of the user's antenna is positioned 1.55 m off the ground, reflecting the height at which a 1.8 m tall person would typically hold the UE. To simulate real-world conditions, we establish six distinct

TABLE I
THE DETAILED SETTINGS FOR 12 SCENARIOS.

Scenarios	Domain type	Dynamic number	Random index	Obstacle
1	Source	No	No	Limited
2	Source	Yes	No	Limited
3	Source	No	Yes	Limited
4	Source	Yes	Yes	Limited
5	Source	No	No	Significant
6	Source	Yes	Yes	Significant
7	Target	No	No	Limited
8	Target	Yes	No	Limited
9	Target	No	Yes	Limited
10	Target	Yes	Yes	Limited
11	Target	No	No	Significant
12	Target	Yes	Yes	Significant



(a) Source domain environment. (b) Target domain environment.

Fig. 11. Experiment environments.

experimental scenarios in the source domain, corresponding to Scenarios 1-6 in Table I. "Dynamic number" means that the number of transmit antennas of each AP is dynamic. "Random index" refers to the indices of the receive antennas on the user being assigned randomly. In the column of "Obstacle", "Limited" indicates that one person occasionally move around in the scene, and "Significant" refers to more than four persons frequently move in the scene. The obstacles in these scenarios could affect the LoS path. Among these scenarios, Scenario 1 represents the commonly used ideal setup in existing studies [11], [14], [21], [37]. Scenarios 2-5 are considered to evaluate the systems' abilities to solve Problem 1 and Problem 2. Scenario 6, on the other hand, simulates a practical large-scale indoor environment with various influencing factors, offering a comprehensive evaluation of GraphFi's performance. We collect 300 WiFi packets at each RP in Scenario 1, with 240 used for training and 60 for testing. In Scenarios 2-6, 60 WiFi packets are collected at each RP for testing. After completing the source domain data collection, we collect the target domain data 15 days later. As shown in Figure 11(b), during these 15 days, the office is used as usual, causing changes in the layout of various objects in the environment (e.g., desk arrangements, chair locations, etc.). To further increase the distribution difference between the source and target domain data, during the target domain data collection, the height of the user's antenna is positioned 1.45 m off the ground, which differs from the height used during the source domain data collection. We also set up six distinct experimental scenarios in the target domain, corresponding to Scenarios 7-12 in Table I. Scenario 7 represents the ideal setup in the target domain, used to evaluate systems' ability to solve Problem 3. Scenarios 8-11 are used to evaluate the

TABLE II
ARCHITECTURES OF ALL APPROACHES.

Approaches		Architectures
GraphFi	Feature aggregation GNN	3 graph convolution layers {128, 128, 128}
	Anomaly detection GNN	Encoder (input dim 128): 3 graph convolution layers {64, 64, 32} Decoder (input dim 32): 3 graph convolution layers {64, 64, 128}
	Location estimation GNN	2 graph convolution layers {128, 64} + FC layer {2} + UDA (only in fine-tuning) with 2 FC layers {32, 2}
UDA-GCN (only in fine-tuning)		2 GCN layers {128, 128} + 3 FC layers {64, 32, 2} + UDA with 2 FC layers {32, 2}
GCN		2 GCN layers {128, 128} + 3 FC layers {64, 32, 2}
GraphSAGE		2 GraphSAGE layers {128, 128} + 3 FC layers {64, 32, 2}
ResNet		Input block (Conv 3×3, 64, stride 2) + 4 types of residual blocks {64, 128, 256, 512} (3 repeats each) + FC layer {2}
DNN		5 FC layers {300, 150, 100, 40, 2}

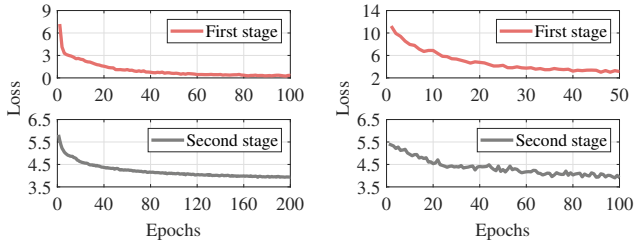
TABLE III
TRAINING/FINE-TUNING SETTINGS FOR ALL APPROACHES.

Approaches	Process type	Optimizers	Epochs	Learning rates	Loss function
GraphFi	First training stage	Adam	100	0.003	$\mathcal{L}^{\mathrm{L}}$
	Second training stage	Adam	200	0.001	$\mathcal{L}^{\mathrm{A}}$
	First fine-tuning stage	Adam	50	0.001	$\mathcal{L}$
	Second fine-tuning stage	Adam	100	0.0004	$\mathcal{L}^{\mathrm{A}}$
UDA-GCN	Fine-tuning	Adam	100	0.0001	$\mathcal{L}$
GCN	Training	Adam	300	0.005	$\mathcal{L}^{\mathrm{L}}$
GraphSAGE	Training	Adam	300	0.005	$\mathcal{L}^{\mathrm{L}}$
ResNet	Training	SGD	50	0.01	$\mathcal{L}^{\mathrm{L}}$
DNN	Training	Adam	150	0.002	$\mathcal{L}^{\mathrm{L}}$

systems' abilities to solve Problems 1 and 2 in the target domain. Scenario 12 simulates a practical indoor environment in the target domain with various influencing factors, used to assess the overall performance in the target domain. We collect 80 WiFi packets at each RP in Scenario 7, with 20 used for GraphFi's fine-tuning and 60 for testing. It is worth noting that the location labels of WiFi packets are not required during fine-tuning. In Scenarios 8-12, 60 WiFi packets are collected at each RP for testing.

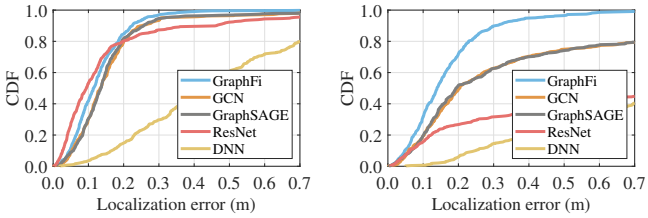
B. Single-Domain Performance

We first assess the systems' overall localization performance in the single-domain setting, which means training and testing using data with location labels solely from the source domain. All considered approaches are trained in Scenario 1 and then tested across Scenarios 1-6. LE is calculated by Eqn. (13), and the average LE is used to quantify the localization performance. To demonstrate the superiority, we compare GraphFi with four learning-based baseline systems: GCN-based [29], GraphSAGE-based [30], ResNet-based [22], and DNN-based [21] systems. Modeling-based algorithms are not considered because they require calculating intermediate quantities such as AoA and ToF. This necessitates knowing the cor-



(a) Training. (b) Fine-tuning.

Fig. 12. Loss curves of GraphFi during the training and fine-tuning stages.

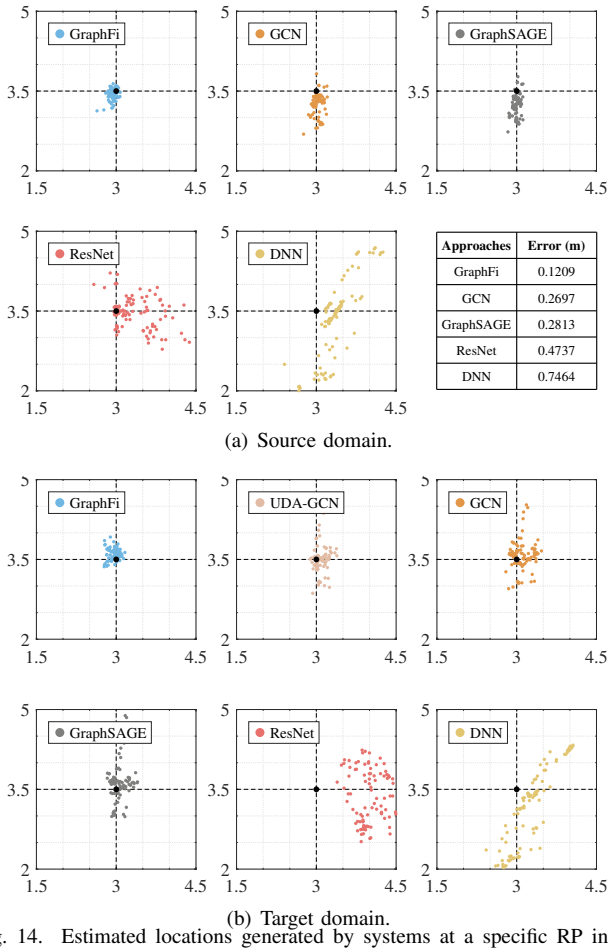


(a) Scenario 1. (b) Scenario 6.

Fig. 13. LE for five systems in Scenarios 1 and 6 of the source domain.

responding CSI for every TX-RX antenna pair and the spacing between neighboring antennas. But Problem 1 would prevent the acquisition of such information, disabling the modeling-based localization algorithms. To achieve localization, GCN-based and GraphSAGE-based systems convert the CSI data into graphs with APs regarded as nodes, the ResNet-based system transforms the CSI data into matrices, and the DNN-based system reshapes the CSI data into vectors. It is important to note that dynamic antenna resource allocation can alter the dimensionality of the CSI data. Since these four baselines can only handle input data with fixed dimensionality, to ensure proper functioning of these models, we apply zero-padding to accommodate the changes in CSI dimensionality. This maintains consistent input dimensionality across all baselines. Moreover, the architectural configurations of GraphFi and the four baselines are presented in Table II, and their training settings are presented in Table III.

First, we show the loss curves for GraphFi's ternary GNNs during the training stages in Figure 12(a). One can clearly see that the model converges rapidly during both the training stages. Then, the localization performance of the five systems is tested across Scenarios 1-6 of the source domain, as shown in Figure 13. It can be observed that the LEs of GraphFi, GCN-based, GraphSAGE-based, and ResNet-based systems are all small in Scenario 1, with the average errors of 0.1293 m, 0.1743 m, 0.1730 m, and 0.1704 m, respectively. This is expected, as the CSI samples in the training and test sets have similar distributions. The high LE of the DNN-based system is likely due to its simple structure and limited expressive power. In Scenario 6, GraphFi maintains a low average LE of 0.1692 m. In contrast, the LEs of GCN-based, GraphSAGE-based, ResNet-based, and DNN-based systems increase markedly, reaching 0.5164 m, 0.5198 m, 1.1893 m, and 1.2051 m, respectively. These increases are attributed to the lack of specific components in these four baselines to address the practical problems in real-world scenarios. This result demonstrates that GraphFi effectively mitigates these problems with minimal performance degradation. To provide a more intuitive comparison of the localization performance

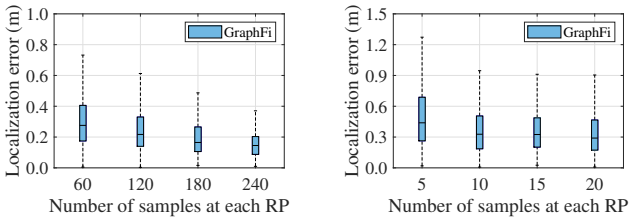


Approaches	Error (m)
GraphFi	0.1209
GCN	0.2697
GraphSAGE	0.2813
ResNet	0.4737
DNN	0.7464

(a) Source domain.

(b) Target domain.

Fig. 14. Estimated locations generated by systems at a specific RP in the source domain and the target domain (unit: meter).



(a) Training. (b) Fine-tuning.

Fig. 15. Localization performance with the number of training and fine-tuning samples at each RP.

across all approaches, Figure 14(a) presents the estimated location visualizations for Scenario 6 of the source domain. The results clearly show that the estimated locations generated by GraphFi are closer to the ground-truth location than those of all other baselines, verifying the effectiveness of the proposed solutions. Moreover, we also illustrate the relationship between LE and the size of the training dataset in Figure 15(a). It is observed that the average LE of GraphFi in Scenario 6 of the source domain gradually decreases as the number of training samples increases. This is because a larger training set enables GraphFi to better learn the mapping between CSI and location coordinates, thereby improving localization accuracy.

To thoroughly demonstrate the effectiveness of GraphFi in the single-domain setting, we present the LEs of the five systems across Scenarios 1-6 of the source domain in Figure 16. It is evident that GraphFi consistently maintains a low LE across different scenarios. Specifically, GraphFi

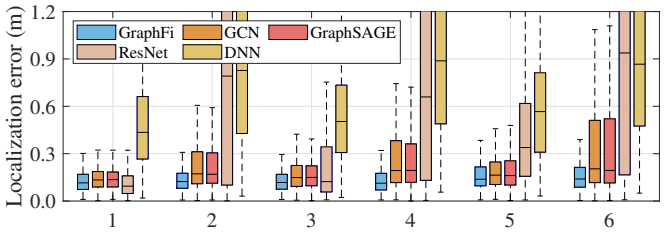
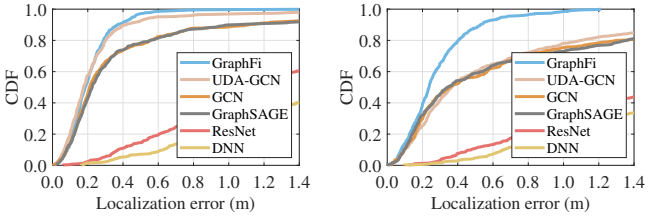


Fig. 16. Localization performance of five systems across Scenarios 1-6.



(a) Scenario 7. (b) Scenario 12.

Fig. 17. LE for six systems in Scenario 7 and 12 of the target domain.

effectively handles dynamic changes in antenna configurations in Scenarios 2, 3, and 4, thanks to our graph construction method, which fully leverages the scalability and permutation invariance of GNNs. Meanwhile, GraphFi performs effectively despite influence from obstacles, benefiting from the proposed anomaly detection GNN. For GCN-based and GraphSAGE-based systems, LEs in Scenarios 2, 4, and 5 are notably higher compared to Scenario 1. This is because these systems treat APs as nodes without accounting for more granular features among antennas. The increased errors in Scenario 5 are particularly due to their failure to consider obstacle influence.

An interesting observation is that the ResNet-based system exhibits the lowest median LE in Scenario 1. This is mainly due to ResNet's deep structure and effective residual learning, which enable it to more effectively extract and utilize complex features from CSI. However, the ResNet-based system adopts a coarse-grained approach by treating the CSI data from all APs as a single entity, which may lead to its high LE in Scenarios 2, 3, and 4. Additionally, the ResNet-based system performs poorly in Scenario 5 due to its inability to handle influence from obstacles. Furthermore, the DNN-based system exhibits high LEs across all scenarios due to its simple structure.

C. Cross-Domain Performance

After evaluating the single-domain performance of GraphFi, we assess its cross-domain capability. To provide a more thorough and accurate evaluation, we introduce an additional baseline, UDA-GCN [50], alongside the four existing baselines in Section VII-B. UDA-GCN is an unsupervised graph domain adaptation method that facilitates knowledge transfer from the source domain to the target domain using a dual graph convolutional network and GRL. In the evaluation, UDA-GCN can be viewed as a GCN-based system with domain adaptation capabilities. Similar to GCN-based and GraphSAGE-based systems, UDA-GCN converts the CSI data into graphs with APs as nodes, and the data processing approach is also similar. In the evaluation of cross-domain capability, labeled data from Scenario 1 of the source domain and a small amount of unlabeled data from Scenario 7 of the target domain are available. First, all the considered approaches are pre-trained using labeled data from Scenario 1 of the source domain. The data from the target domain, however, is unlabeled. Since the GCN-based, GraphSAGE-based, ResNet-based, and DNN-based localization systems lack mechanisms for handling unlabeled data, they cannot make use of the unlabeled data from Scenario 7. In contrast, GraphFi and UDA-GCN, with their domain adaptation mechanisms, are able to effectively leverage this unlabeled data for fine-tuning. Moreover, the architectural configurations of GraphFi and UDA-GCN are shown in Table II, and their fine-tuning settings are provided in Table III.

First of all, the loss curves of GraphFi's ternary GNNs during the fine-tuning stage in Figure 12(b) demonstrate its fast convergence. Then, we test the six systems across Scenarios 7-12 of the target domain, and the overall performance of six systems in the ideal scenario (i.e., Scenario 7) and practical scenario (i.e., Scenario 12) of the target domain is presented in Figure 17. In Scenario 7, the average LEs for GraphFi, UDA-GCN, GCN-based, GraphSAGE-based, ResNet-based, and DNN-based systems are 0.2145 m, 0.2624 m, 0.4530 m, 0.4595 m, 1.4899 m, and 1.8038 m, respectively. The results in Scenario 7 reflect the systems' ability to address Problem 3. GraphFi and UDA-GCN, benefiting from the domain adaptation mechanisms, are able to fine-tune using a small number of unlabeled target domain samples and extract domain-invariant features. As a result, they can adapt to the data distribution of the target domain and maintain relatively low LEs. On the other hand, GCN-based, GraphSAGE-based, ResNet-based, and DNN-based systems, which are unable to process the unlabeled data from the target domain, can only make predictions based on the data distribution of the source domain. This leads to their inability to capture the data distribution of the target domain, causing their LEs in Scenario 7 to increase significantly compared to Scenario 1. In Scenario 12, the average LEs for GraphFi, UDA-GCN, GCN-based, GraphSAGE-based, ResNet-based, and DNN-based systems are 0.2851 m, 0.7177 m, 0.8008 m 0.8132 m, 1.7973 m, and 1.9707 m, respectively. GraphFi is able to maintain a relatively low LE in Scenario 12. This indicates that GraphFi not only exhibits efficient domain transfer capability with low cost, but also effectively addresses the practical challenges present in the real-world scenario of the target domain. Due to reasons similar to those mentioned in Section VII-B, the GCN-based, GraphSAGE-based, ResNet-based, and DNN-based systems perform poorly in practical scenarios. Additionally, due to domain discrepancies, the LEs of these algorithms in Scenario 12 become larger compared to Scenario 6. For UDA-GCN, although its LE in Scenario 7 is similar to that of GraphFi, its LE in Scenario 12 is significantly higher. This indicates that although UDA-GCN can mitigate the impact of domain shifts on LE through domain adaptation mechanisms, it lacks the necessary components to effectively address Problems 1 and 2 mentioned in Section III-A. Consequently, its performance in the real-world scenario of the target domain is suboptimal. The above results preliminarily suggest that, through the close collaboration of its components, GraphFi is

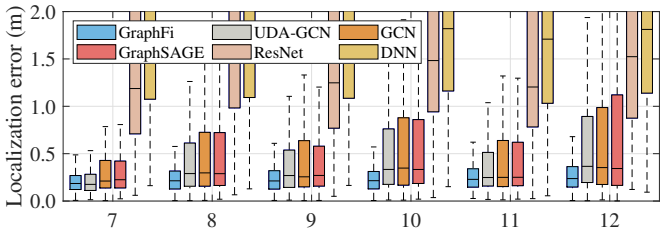


Fig. 18. Localization performance of six systems across Scenarios 7-12.

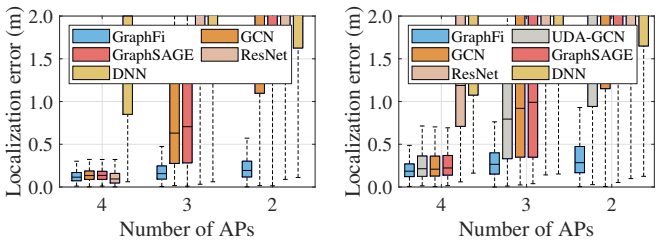
capable of addressing Problems 1, 2, and 3 simultaneously. To provide a more intuitive comparison across all approaches, Figure 14(b) presents the estimated location visualizations for Scenario 12 of the target domain. The results clearly show that the estimated locations generated by GraphFi are closer to the ground-truth location than those of all other baselines, verifying the effectiveness of the proposed solutions. Moreover, we also illustrate the relationship between LE and the size of the fine-tuning dataset in Figure 15(b). It is observed that the average LE of GraphFi in Scenario 12 of the target domain gradually decreases as the number of fine-tuning samples increases. This is because a larger fine-tuning dataset allows GraphFi to capture domain-invariant features more effectively, enhancing the accuracy of location estimation.

To thoroughly demonstrate the effectiveness of GraphFi in the cross-domain setting, we present the LEs of the six systems across Scenarios 7-12 of the target domain in Figure 18. The results indicate that GraphFi maintains relatively low LEs across all six scenarios in the target domain. By comparing GraphFi's LEs in Scenario 7 with those in Scenarios 8, 9, and 10, it is evident that the differences are minimal. This suggests that, after domain adaptation, our graph construction method remains effective, enabling GraphFi to address dynamic changes in antenna configurations in the target domain. The result in Scenario 11 demonstrates that GraphFi can effectively handle influence from obstacles in the target domain. By utilizing a small amount of unlabeled data from Scenario 7, the anomaly detection GNN learns the distribution of normal data, which allows it to detect anomalous data in Scenario 11. Due to the reasons mentioned in Section VII-B and the domain shifts, the GCN-based, GraphSAGE-based, ResNet-based, and DNN-based systems have larger LEs in Scenarios 8-11.

It is worth noting that UDA-GCN's LEs in Scenarios 8-11 are slightly lower than those of the GCN-based and GraphSAGE-based systems, but still significantly higher than those of GraphFi. This indicates that the presence of domain adaptation mechanisms allows UDA-GCN to mitigate the performance degradation caused by domain shifts. However, due to the lack of corresponding mechanisms, UDA-GCN is unable to address the dynamic antenna configuration problems and influence from obstacles in the target domain. This indicates that our GraphFi also possesses irreplaceable advantages in the target domain.

D. Effect of AP Quantity on Localization Performance

First, we construct three test sets in the source domain, with the numbers of available APs being 4, 3, and 2, respectively. To control for confounding factors, these three test sets are configured identically to the training set except for the number



(a) Source domain. (b) Target domain.

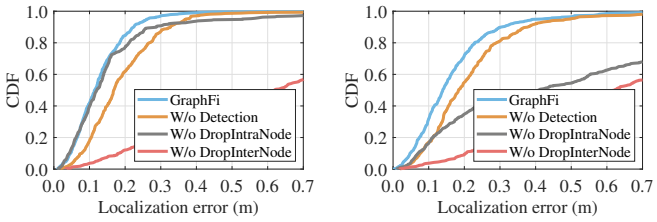
Fig. 19. Localization performance of systems in the source and target domain with varying numbers of APs.

of APs. Figure 19(a) presents the LEs of five systems on these test sets. It can be observed that the LEs of all systems increase as the number of APs decreases. However, compared to other baselines, GraphFi maintains relatively stable localization performance under varying AP conditions. This is primarily due to the use of data augmentation in GraphFi's ternary GNNs, as introduced in Section VI-D. Similar to a conventional dropout layer, this GNN-specific augmentation technique randomly drops some nodes within a portion of the inter-AP graphs. Since there is a one-to-one correspondence between inter-AP graphs and APs, this data augmentation effectively equips ternary GNNs with the ability to adapt to dynamic changes in the number of APs, thereby enabling GraphFi to maintain its localization performance even when some APs go offline.

Similarly, we construct three test sets in the target domain, with the numbers of available APs being 4, 3, and 2, respectively. As in the source domain, the number of APs is the only varying factor in these target-domain test sets. Figure 19(b) presents the LEs of six systems on these test sets. It can be observed that the LEs of all systems increase as the number of APs decreases. However, compared to the other baselines, GraphFi maintains superior localization performance under varying AP conditions. These experimental results demonstrate that GraphFi is capable of adapting to dynamic changes in the number of APs in both the source and target domains.

E. Ablation Study

To evaluate the effectiveness of the tailored components in GraphFi, including the anomaly detection function, two data augmentations and the UDA module, we conduct ablation experiments. In these experiments, we evaluate performance by sequentially removing each component from GraphFi. Specifically, we consider four baselines: one without the anomaly detection component, one without dropping applied to nodes in the intra-AP graph for data augmentation (i.e., "DropIntraNode"), one without dropping applied to nodes in the inter-AP graph for data augmentation (i.e., "DropInterNode") and one without the UDA module (i.e., "UDA"). By comparing the first three baselines with the complete GraphFi in the source domain, we can clearly identify the performance gains contributed by the anomaly detection, "DropIntraNode", and "DropInterNode" components. By comparing the fourth baseline with the complete GraphFi in the target domain, we can identify the ability of the UDA module to achieve domain adaptation with a small number of unlabeled samples.



(a) Scenario 1. (b) Scenario 6.

Fig. 20. CDF of LE for four schemes in Scenario 1 and Scenario 6 in the ablation experiment of the source domain.

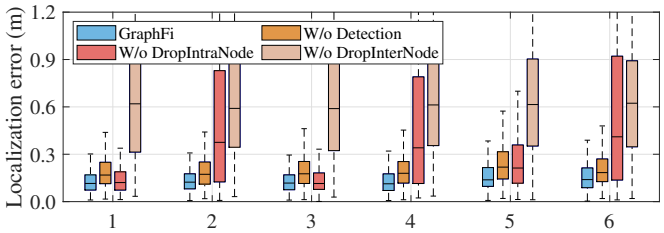
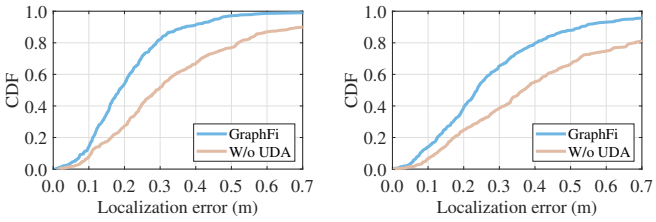


Fig. 21. Localization performance of four schemes across Scenarios 1-6 in the ablation experiment of the source domain.



(a) Scenario 7. (b) Scenario 12.

Fig. 22. CDF of LE for two schemes in Scenario 7 and Scenario 12 in the ablation experiment of the target domain.

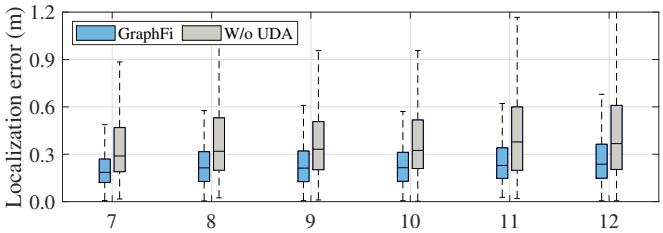


Fig. 23. Localization performance of two schemes across Scenarios 7-12 in the ablation experiment of the target domain.

First, we evaluate the contributions of the anomaly detection, "DropIntraNode," and "DropInterNode" components in the source domain. The overall performance of the four schemes (i.e., GraphFi, "W/o Detection", "W/o DropIntraNode", and "W/o DropInterNode") in Scenarios 1 and 6 is presented in Figure 20. In Scenario 1, the LEs for all schemes except "W/o DropInterNode" are low due to both the simplicity of the scenario and the fact that all models are trained using CSI samples from this scenario. However, in Scenario 6, a more practical real-world setting, the LE of GraphFi is significantly lower than that of the other three baselines. This preliminary result confirms the effectiveness of each component in GraphFi. To further validate the contribution of each component, we also present the performance of the four schemes across Scenarios 1-6 in Figure 21. As we noted earlier, these scenarios vary in their settings, as detailed in Table I. The performance of the scheme "W/o DropIntraNode" is significantly worse than that of GraphFi in Scenarios 2, 4, and 6, but similar in Scenarios 1, 3, and 5. This discrepancy arises because Scenarios 2, 4, and 6 involve dynamic antenna configurations, while Scenarios 1, 3, and 5 do not. Although the scalability of GNNs allows them to handle graphs with varying numbers of nodes, this capability is not fully realized without an appropriate training strategy. The "DropIntraNode" in GraphFi enables it to adapt to different node counts, effectively utilizing GNN scalability and addressing dynamic antenna resource allocation. Similarly, the scheme "W/o DropInterNode" shows higher LEs compared to GraphFi across Scenarios 1-6. This can be explained as follows: during training, all the nodes of the inter-AP graph are used for localization. However, in the test stage, some nodes in the inter-AP graph may be affected by obstacles, and the anomaly detection GNN will filter out these nodes. Consequently, the scheme "W/o DropInterNode" fails to adapt to the varying number of nodes in inter-AP graphs and performs poorly. Additionally, GraphFi consistently demonstrates lower LEs compared to the scheme "W/o Detection" across Scenarios 1-6, particularly in Scenarios 5 and 6 where obstacle influence is significant. All the above results demonstrate that the proposed anomaly detection GNN effectively handles obstacle influence by filtering out obstacle-affected CSI data.

We then evaluate the contribution of the UDA module in the target domain. The overall performance of the two schemes (i.e., GraphFi and "W/o UDA") in Scenarios 7 and 12 is presented in Figure 22. It can be observed that in Scenarios 7 and 12, the LEs of GraphFi are lower than those of the scheme "W/o UDA". To further validate the effectiveness of the UDA module, we also present the performance of the two schemes across Scenarios 7-12 in Figure 23. It can be observed that the localization performance of GraphFi outperforms that of "W/o UDA" in Scenarios 7-12. This clearly demonstrates the critical role of the UDA module in enabling domain adaptation for GraphFi, preventing a decline in localization performance caused by domain shift.

In summary, all the above-mentioned findings underscore the effectiveness of each component within GraphFi. Moreover, the careful designs ensure that these components work synergistically, allowing GraphFi to achieve low LEs and high robustness across various scenarios.

VIII. Conclusion

This paper introduces GraphFi, a cutting-edge WiFi-based active indoor localization system. It features two innovative graph structures: the intra-AP graph, which captures local features from antennas within each AP, and the inter-AP graph, which integrates global features across all available APs. These graphs effectively handle dynamic antenna configuration problems. Utilizing two customized GNNs, GraphFi accurately determines user location based on these graphs. Additionally, to address the problem of unpredictable obstacle influence, GraphFi incorporates an anomaly detection mechanism to filter out obstacle-affected features prior to location estimation. Furthermore, GraphFi integrates a GRL-based UDA method to maintain localization performance in a cost-efficient manner. Real-world experiments show that GraphFi achieves average LEs of 0.17 m in a single-domain setting and 0.2851 m in

a cross-domain setting. As the first study to leverage GNNs for addressing dynamic antenna configuration and obstacle influence in WiFi indoor localization, our work marks a significant step toward practical WiFi-based localization.

Nevertheless, several promising directions remain for future exploration. First, due to experimental constraints, we are currently unable to collect large-scale data in diverse environments. We plan to address this in future work by deploying our system in more complex and realistic settings, such as airports and shopping malls. In addition, while GraphFi effectively handles inconsistent data distributions caused by sudden obstacles, its applicability to diverse environments could be improved, and incorporating meta-learning [51]–[53] offers an effective means. Furthermore, this work addresses the mitigation of obstacle-affected APs with a limited number of APs. When the AP count is large, employing all APs for localization can be computationally expensive; selecting an optimal subset of APs can effectively reduce this cost [54], which we leave for future work.

REFERENCES

[1] F. Zafari, A. Gkelias, and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2568–2599, 3rd Quart. 2019.

[2] Z. Chen, G. Zhu, S. Wang, Y. Xu, J. Xiong, J. Zhao, J. Luo, and X. Wang, "M3: Multipath Assisted Wi-Fi Localization with a Single Access Point," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 588–602, Mar. 2019.

[3] J. Hu, Z. Chen, T. Zheng, R. Schober, and J. Luo, "HoloFed: Environment-Adaptive Positioning via Multi-Band Reconfigurable Holographic Surfaces and Federated Learning," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3736–3751, Dec. 2023.

[4] X. Li, H. Wang, Z. Chen, Z. Jiang, and J. Luo, "UWB-Fi: Pushing Wi-Fi towards Ultra-wideband for Fine-Granularity Sensing," in *Proc. ACM MobiSys*, Jun. 2024, pp. 42–55.

[5] A. Poulose and D. S. Han, "UWB Indoor Localization Using Deep Learning LSTM Networks," *Appl. Sci.*, vol. 10, no. 18, p. 6290, Sep. 2020.

[6] T. Koike-Akino, P. Wang, M. Pajovic, H. Sun, and P. V. Orlik, "Fingerprinting-Based Indoor Localization With Commercial MMWave WiFi: A Deep Learning Approach," *IEEE Access*, vol. 8, pp. 84 879–84 892, Apr. 2020.

[7] M. Hazas and A. Hopper, "Broadband Ultrasonic Location Systems for Improved Indoor Positioning," *IEEE Trans. Mobile Comput.*, vol. 5, no. 5, pp. 536–547, May 2006.

[8] Y. He, G. Yu, Y. Cai, and H. Luo, "Integrated Sensing, Computation, and Communication: System Framework and Performance Optimization," *IEEE Trans. Wireless Commun.*, vol. 23, no. 2, pp. 1114–1128, Feb. 2024.

[9] Y. He, J. Liu, M. Li, G. Yu, J. Han, and K. Ren, "SenCom: Integrated Sensing and Communication with Practical WiFi," in *Proc. ACM MobiCom*, Oct. 2023, pp. 1–16.

[10] Y. He, J. Liu, M. Li, G. Yu, and J. Han, "Forward-Compatible Integrated Sensing and Communication for WiFi," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2440–2456, Sept. 2024.

[11] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "SpotFi: Decimeter Level Localization Using WiFi," in *Proc. ACM SIGCOMM*, Aug. 2015, pp. 269–282.

[12] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-Level Localization with a Single WiFi Access Point," in *Proc. USENIX NSDI*, Mar. 2016, pp. 165–178.

[13] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep Learning for Indoor Fingerprinting Using Channel State Information," in *Proc. IEEE WCNC*, Mar. 2015, pp. 1666–1671.

[14] X. Wang, X. Wang, and S. Mao, "CiFi: Deep Convolutional Neural Networks for Indoor Localization with 5 GHz Wi-Fi," in *Proc. IEEE ICC*, May 2017, pp. 1–6.

[15] S. Yang, D. Zhang, R. Song, P. Yin, and Y. Chen, "Multiple WiFi Access Points Co-localization through Joint AOA Estimation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 2, pp. 1488–1502, Feb. 2023.

[16] T. F. Sanam and H. Godrich, "A Multi-View Discriminant Learning Approach for Indoor Localization using Amplitude and Phase Features of CSI," *IEEE Access*, vol. 8, pp. 59 947–59 959, Mar. 2020.

[17] L. Wang and S. Pasricha, "A Framework for CSI-Based Indoor Localization with ID Convolutional Neural Networks," in *Proc. IEEE IPIN*, Sept. 2022, pp. 1–8.

[18] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Impact of IEEE 802.11n/ac PHY/MAC High Throughput Enhancements on Transport and Application Protocols—A Survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2050–2091, 4th Quart. 2017.

[19] Cisco, "Aironet 1850 Series Access Points Data Sheet," https://www.cisco.com/c/en/us/products/collateral/wireless/aironet-1850-series-access-points/datasheet-c78-734256.html, 2021, online; accessed: 16 November 2025.

[20] EnGenius, "EnGenius AP Configuration Best Practices," https://doctorengenius.engeniustech.com/en/articles/6708818-engenius-ap-configuration-best-practices, 2023, online; accessed: 16 November 2025.

[21] G.-S. Wu and P.-H. Tseng, "A Deep Neural Network-based Indoor Positioning Method using Channel State Information," in *Proc. IEEE ICNC*, Mar. 2018, pp. 290–294.

[22] X. Wang, X. Wang, and S. Mao, "Indoor Fingerprinting with Bimodal CSI Tensors: A Deep Residual Sharing Learning Approach," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4498–4513, Mar. 2020.

[23] X. Zhu, W. Qu, X. Zhou, L. Zhao, Z. Ning, and T. Qiu, "Intelligent Fingerprint-based Localization Scheme using CSI Images for Internet of Things," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2378–2391, Jul.-Aug. 2022.

[24] Z. Jiang, "Characterizing Jump Discontinuities in CSI Amplitude Data," https://gitlab.com/wifisensing/picoscenes-issue-tracker/-/issues/102, 2022, online; accessed: 16 November 2025.

[25] ——, "Amplitude Discontinuities Observed in Intel AX200/AX210 Wi-Fi NICs," https://gitlab.com/wifisensing/picoscenes-issue-tracker/-/issues/216, 2024, online; accessed: 16 November 2025.

[26] ——, "Problems Observed in Data Collected Using the Intel AX210," https://gitlab.com/wifisensing/picoscenes-issue-tracker/-/issues/222, 2024, online; accessed: 16 November 2025.

[27] R. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Trans. Antennas Propag.*, vol. 34, no. 3, pp. 276–280, Mar. 1986.

[28] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[29] Y. Sun, Q. Xie, G. Pan, S. Zhang, and S. Xu, "A Novel GCN based Indoor Localization System with Multiple Access Points," in *Proc. IEEE IWCMC*, Jun. 2021, pp. 9–14.

[30] X. Luo and N. Meratnia, "A Geometric Deep Learning Framework for Accurate Indoor Localization," in *Proc. IEEE IPIN*, Sept. 2022, pp. 1–8.

[31] R. Ayyalasomayajula, A. Arun, C. Wu, S. Sharma, A. R. Sethi, D. Vasisht, and D. Bharadia, "Deep Learning based Wireless Localization for Indoor Navigation," in *Proc. ACM MobiCom*, Apr. 2020, pp. 1–14.

[32] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain Generalization: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4396–4415, Apr. 2023.

[33] S. Liu, Z. Chen, M. Wu, C. Liu, and L. Chen, "WiSR: Wireless Domain Generalization Based on Style Randomization," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4520–4532, May 2024.

[34] J. Wang, J. Xiong, H. Jiang, K. Jamieson, X. Chen, D. Fang, and C. Wang, "Low Human-Effort, Device-Free Localization with Fine-Grained Subcarrier Information," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2550–2563, Nov. 2018.

[35] S. Han, Y. Li, W. Meng, C. Li, T. Liu, and Y. Zhang, "Indoor Localization With a Single Wi-Fi Access Point Based on OFDM-MIMO," *IEEE Syst. J.*, vol. 13, no. 1, pp. 964–972, Mar. 2019.

[36] L. Zhang, Q. Gao, X. Ma, J. Wang, T. Yang, and H. Wang, "DeFi: Robust Training-Free Device-Free Wireless Localization with WiFi," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8822–8831, Sept. 2018.

[37] Y. Xie, J. Xiong, M. Li, and K. Jamieson, "mD-Track: Leveraging Multi-Dimensionality for Passive Indoor Wi-Fi Tracking," in *Proc. ACM MobiCom*, Oct. 2019, pp. 1–16.

[38] J. Hu, D. Niyato, and J. Luo, "Cross-Domain Learning Framework for Tracking Users in RIS-Aided Multi-Band ISAC Systems With Sparse Labeled Data," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 10, pp. 2754–2768, Oct. 2024.

[39] P. Li, H. Cui, A. Khan, U. Raza, R. Piechocki, A. Doufexi, and T. Farnham, "Wireless Localisation in WiFi using Novel Deep Architectures," in *Proc. IEEE ICPR*, Jan. 2021, pp. 6251–6258.

[40] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge university press, 2005.
[41] Y. He, X. Li, and J. Luo, "Task-Oriented Integrated Sensing and Semantic Communications for Multi-Device Video Analytics," *IEEE Trans. Mobile Comput.*, 2025, under review.
[42] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[43] Y. Ganin and V. Lempitsky, "Unsupervised Domain Adaptation by Backpropagation," in *Proc. ACM ICML*, Jul. 2015, pp. 1180–1189.
[44] I. S. Association, "IEEE Std 802.11-2016, IEEE Standard for Local and Metropolitan Area Networks—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 2016.
[45] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. ACM ICML*, Jul. 2015, pp. 448–456.
[46] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proc. ACM ICML*, Jun. 2010, pp. 807–814.
[47] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep Anomaly Detection on Attributed Networks," in *Proc. SIAM SDM*, May 2019, pp. 594–602.
[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[49] Z. Jiang, T. H. Luan, X. Ren, D. Lv, H. Hao, J. Wang, K. Zhao, W. Xi, Y. Xu, and R. Li, "Eliminating the Barriers: Demystifying Wi-Fi Baseband Design and Introducing the PicoScenes Wi-Fi Sensing Platform," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4476–4496, Mar. 2021.
[50] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, "Unsupervised Domain Adaptive Graph Convolutional Networks," in *Proc. ACM WWW*, Apr. 2020, pp. 1457–1467.
[51] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proc. ACM ICML*, Aug. 2017, pp. 1126–1135.
[52] J. Gao, D. Wu, F. Yin, Q. Kong, L. Xu, and S. Cui, "MetaLoc: Learning to Learn Wireless Localization," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3831–3847, Dec. 2023.
[53] Q. Xiao, Z. Ye, Y. He, J. Liu, and G. Yu, "Meta-SimGNN: Adaptive and Robust WiFi Localization Across Dynamic Configurations and Diverse Scenarios," *IEEE Trans. Commun.*, 2025, early access, doi: 10.1109/TCOMM.2025.3637041.
[54] S. Wang, S. Zhang, J. Ma, and O. A. Dobre, "Graph-Neural-Network-Based WiFi Indoor Localization System With Access Point Selection," *IEEE Internet Things J.*, pp. 33 550–33 564, Oct. 2024.