Automatic Update for Wi-Fi Fingerprinting Indoor Localization via Multi-Target Domain Adaptation

JIANKUN WANG, College of Intelligence and Computing, Tianjin University, China
ZENGHUA ZHAO*, College of Intelligence and Computing, Tianjin University, China
MENGLING OU, College of Intelligence and Computing, Tianjin University, China
JIAYANG CUI, College of Intelligence and Computing, Tianjin University, China
BIN WU, College of Intelligence and Computing, Tianjin University, China

Wi-Fi fingerprinting system in the long term suffers from gradually deteriorative localization accuracy, leading to poor user experiences. To keep high accuracy yet at a low cost, we first study long-term variation of access points (APs) and characteristics of their Wi-Fi signals through over-one-year experiments. Motivated by the experimental findings, we then design MTLoc, a **M**ulti-**T**arget domain adaptation network-based Wi-Fi fingerprinting **Loc**alization system. As the core, MTDAN (**M**ulti-**T**arget **D**omain **A**daptation **N**etwork) model adopts the framework of generative adversarial network to learn time-invariant, time-specific, and location-aware features from the source and target domains. To enhance the alignment among the source and targets, two-level cycle consistency constraints are proposed. Hence, MTDAN is able to transfer location knowledge from the source domain to multiple targets. In addition, domain selection and outlier detection are designed to avoid explosive growth of storage for targets and to limit the impact of random variations of Wi-Fi signals. Extensive experiments are carried out on five datasets collected over two years in various real-world indoor environments with a total area of $8,350\,\text{m}^2$. Experimental results demonstrate that MTLoc retains high localization accuracy with limited storage and training cost in the long term, which significantly outperforms its counterparts. *We share our dataset to the community for other researchers to validate our results and conduct further research.*

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing theory, concepts and paradigms**; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: indoor localization, Wi-Fi fingerprinting, fingerprint update, unsupervised multi-target domain adaptation, deep learning

ACM Reference Format:
Jiankun Wang, Zenghua Zhao, Mengling Ou, Jiayang Cui, and Bin Wu. 2023. Automatic Update for Wi-Fi Fingerprinting Indoor Localization via Multi-Target Domain Adaptation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 78 (June 2023), 27 pages. https://doi.org/10.1145/3596239

*Corresponding author.

Authors' addresses: Jiankun Wang, jiankunwang@tju.edu.cn, College of Intelligence and Computing, Tianjin University, 135 Yaguan Rd, Jinnan Dist, Tianjin, China; Zenghua Zhao, zenghua@tju.edu.cn, College of Intelligence and Computing, Tianjin University, 135 Yaguan Rd, Jinnan Dist, Tianjin, China; Mengling Ou, oumengling@tju.edu.cn, College of Intelligence and Computing, Tianjin University, 135 Yaguan Rd, Jinnan Dist, Tianjin, China; Jiayang Cui, jycui@tju.edu.cn, College of Intelligence and Computing, Tianjin University, 135 Yaguan Rd, Jinnan Dist, Tianjin, China; Bin Wu, binw@tju.edu.cn, College of Intelligence and Computing, Tianjin University, 135 Yaguan Rd, Jinnan Dist, Tianjin, China.

2474-9567/2023/6-ART78 $15.00
https://doi.org/10.1145/3596239

1 INTRODUCTION

Due to pervasive deployments of access points (APs) and ubiquities of Wi-Fi-enabled mobile devices, Wi-Fi fingerprinting indoor localization has attracted much attention from both academia and industry in the past decade [35]. However, in practice Wi-Fi fingerprinting system in the long term suffers from gradually deteriorative localization accuracy, leading to poor user experiences. The primary reason is the mismatching between the varied Wi-Fi signals and the stale fingerprint database [9, 24].

Generally, it is costly to keep high localization accuracy in the long term. One straightforward way is updating the fingerprint database periodically by site survey, but it is labor-intensive and time-consuming. To reduce the cost, crowdsourcing is utilized to collect fingerprints without site survey [48]. Unfortunately, crowdsourced fingerprints do not have location information. In order to associate crowdsourced fingerprints with locations, many researches leverage inertial sensors or floorplans to assist localizing, yet introducing extra costs [29, 41, 47] . The state-of-the-art iToLoc [18] breaks through the cost barrier by exploiting deep learning. Although iToLoc can keep a high accuracy in a short time, it approaches only room-level accuracy in the long run.

In this paper, we aim to achieve high localization accuracy in the long term yet at a low cost by using crowdsourced fingerprints, with neither inertial sensor assistance nor floorplan constraints. To this end, we first study the long-term variation of APs and characteristics of Wi-Fi signals through comprehensive experiments. Particularly, we collect Wi-Fi fingerprints in an office building on our campus and a shopping mall persistently over one year. According to our observations, we define three types of APs: common APs, semi-common APs, and temporary APs. Common APs are those appearing in the first collection and shared with subsequent collections; semi-common APs are new APs that appear in one collection (except for the first one) and remain in subsequent collections; while temporary APs appear only once or twice throughout the experiment.

By analyzing extensive experiment data, we found that i) in the long term the number of common APs decreases with time, which impacts the localization accuracy significantly; ii) The number of different APs increases gradually with the length of time interval between two collections, where common APs and semi-common APs are bridges connecting the two ends; and iii) Wi-Fi signals of most common APs are strong at many reference points and are stable over a long time.

Motivated by the above findings, we then design MTLoc, a **M**ulti-**T**arget domain adaptation network-based Wi-Fi fingerprinting **Loc**alization system. Specifically, taking each Wi-Fi fingerprint collection as a domain, the first collection is the source domain, and the subsequent collections are target domains. Only the source domain is labeled with locations, i.e., the original fingerprint database. The targets are unlabeled, i.e., the crowdsourced fingerprints. The stability of signals from common APs allows to transfer location knowledge from the source to the targets. Furthermore, the gradual variation of common and semi-common APs among targets makes it possible to transfer from the source to a faraway target through multiple intermediate targets.

MTLoc has two primary modules: MTDAN (**M**ulti-**T**arget **D**omain **A**daptation **N**etwork) and domain management. MTDAN consists of a feature extractor, a generator, a discriminator, and a location predictor under the framework of generative adversarial network (GAN) [6]. Cooperating with the generator and discriminator, the feature extractor learns both time-invariant and time-specific features from the source and multiple targets. Driven by location predictor, feature extractor also learns location-aware features. At the same time, with extracted features location predictor is trained to estimate locations with high accuracy. Moreover, two-level cycle consistency constraints are designed to ensure effective feature extraction. Therefore, the well-trained MTDAN is able to transfer location knowledge from the source to the targets, and thus is used for on-line localization.

The domain management consists of domain selection and outlier detection. While MTLoc runs, Wi-Fi signals for location query in a time period are collected as a new target. The number of new targets increases with time. If all the new targets join model training each time, there will be an explosive growth of storage and training time. Therefore, the domain management selects representative ones from new targets and stores them in a storage pool. Only representative targets in the pool join the training thus saving storage and training time. Actually, one representative target may join trainings for several times to avoid being forgotten. In addition, some new targets may contain varied Wi-Fi signals caused by random events such as body blockage and turning off APs. These targets are very different from others, or outliers. Since an outlier is usually transient, it joins the training only once. As time goes by, the model will forget the outlier thus limiting its impact on the subsequent localization. We design effective approaches to select representative targets and detect outliers. In addition, several training strategies are also designed to update MTDAN efficiently.

To evaluate the performance of MTLoc, we carry out extensive experiments on five datasets collected in real-world scenarios, including an office building, a large shopping mall, and a library, with a total area of $8,350\,m^2$. The time periods covered by the datasets vary from 3 months to 25 months. The dataset from library is open [22]. Others are ours and have been released for further research [1].

The performance of MTLoc is compared with that of the state-of-the-art models or systems: DANN [4], iToLoc [18] and CNNLoc [14]. In the office, the average localization accuracy achieves 1.2 m after 15 months, which is better than the benchmarks by 4.1 m, 5.0 m, and 8.1 m, respectively. In the mall, the accuracy achieves 7.8 m after 13 months, which is much better than the benchmarks by 6.9 m, 18.4 m, and 14.3 m, respectively. In the library, even after two years the accuracy still keeps 2.3 m, which is better than the benchmarks by 1.1 m, 0.9 m, and 1.2 m, respectively. The experimental results show that MTLoc significantly outperforms the benchmarks on all the datasets in the long term. Whereas, the storage and training costs are limited.

In summary, we make the following contributions:

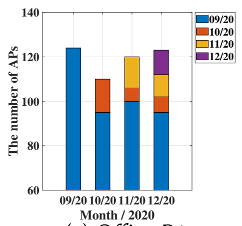
- We carry out an over-one-year experimental study on variation of APs and characteristics of Wi-Fi signals. We found that in the long term the number of common APs impacts the localization accuracy significantly and Wi-Fi signals of most common APs are stable over a long time. To the best of our knowledge, this is the first study on long-term variations of APs and their Wi-Fi signals. We believe the experimental findings will benefit deployment of Wi-Fi fingerprinting localization systems in practice.
- Inspired by the experimental findings, we propose a deep multi-target domain adaptation model MTDAN and design a practical Wi-Fi fingerprinting localization system MTLoc. MTDAN is able to learn time-invariant, time-specific, and location-aware features of long-time Wi-Fi signals, and to transfer location knowledge from the source to the targets. By domain selection and outlier detection, MTLoc reduces storage for targets and is capable of handling random variations of Wi-Fi signals.
- We evaluate MTLoc on five datasets collected in various real-world indoor environments with a total area of $8,350\,\mathrm{m}^2$. Experimental results demonstrate that MTLoc retains high accuracy yet at a low cost in the long run, outperforming the benchmarks significantly.

The rest of the paper is organized as follows. Section 2 describes our long-term experimental study and motivation. We overview MTLoc in Section 3, and present the design of MTLoc in Section 4, 5, and 6. Section 7 evaluates the performance of MTLoc. Limitations and future work are discussed in Section 8. Section 9 reviews the related work, and Section 10 concludes the paper.

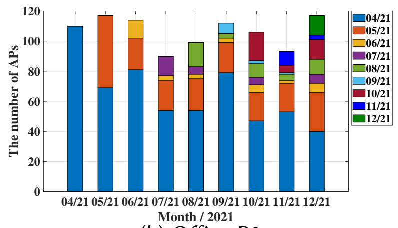
2 LONG-TERM EXPERIMENTAL STUDY AND MOTIVATION

In practice, a long-run Wi-Fi fingerprinting localization system experiences variation of APs. For example, existing APs are removed permanently or turned off for a short time, new APs are deployed. As a result, the number of APs varies over time, leading to variation of the fingerprints. In this section, through comprehensive experiments we study the long-term variation of APs and characteristics of Wi-Fi signals. The insights of the study motivate us to design our MTDAN. We also point out challenges of the design at the end of the section.

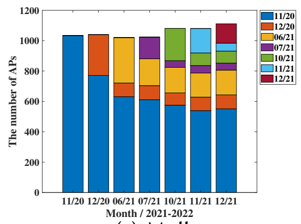
[1]Our dataset is publicly available: https://github.com/WirelessGroupTJU/MTLocData



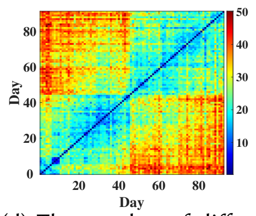
(a) Office P1



(b) Office P2



(c) Mall



(d) The number of different APs on Office P1

Fig. 1. The number of APs varies over time.

2.1 Experimental Study on Long-Term Characteristics of Wi-Fi Fingerprints

We persistently collected Wi-Fi fingerprints in an office building on our campus and a shopping mall over one year. The floorplans of the two sites are shown in Fig. 7. The collections were interrupted from time to time due to holidays or epidemic of coronavirus. Therefore, in the office building the experiments are partitioned into two periods: P1 and P2. For more detail please refer to Sec 7.

We found there are three types of APs: common APs, semi-common APs, and temporary APs. Common APs are heard at the first collection, and keep on showing thereafter; Semi-common APs appear *after* the first collection and keep on showing thereafter; Temporary APs show only once or twice throughout the experiments. By examining their service set identifiers (SSIDs), we found that common APs and semi-common APs are usually infrastructures deployed by the university or Internet service providers, or are personal wireless routers. Whereas most of temporary APs are private hotspots provided by smartphones.

2.1.1 Long-Term Variations of APs. Taking the Wi-Fi data collected at the first time as a benchmark, we study the long-term variation of APs. Fig. 1 plots the number of APs in the first collection and that of APs in subsequent collections with time interval of one month in the form of stacked bars. The number of new APs appearing in a collection is colored differently and keeps the same color thereafter. For example, the second bar is stacked with two colors. The lower one is the number of APs already in the first collection, and the higher one is the number of new APs appearing in current collection.

As shown in Fig. 1, the number of common APs (at the bottom of a bar) generally decreases over time. In office P1 (Fig. 1a), the number of APs decreases from 124 to 95 over three months by 23.4%. For a longer period in P2, the number decreases much more, from 110 to 40, by 63.6%. The same result holds in the mall where the number decreases from 1033 to 551 over one year by 46.7%.

On the other hand, new APs appear at each collection. Apart from the first two, the APs in a collection consist of common APs, semi-common APs, and new APs. Since some of new APs become semi-common APs in subsequent collections, the number of common APs and semi-common APs is much more than that of temporary ones in a collection. As time goes by, the number and component of APs are significantly different from that in the first collection. We illustrate the number of different APs between any two collections in Fig. 1d. It shows that contiguous collections share more common and semi-common APs than those far away from each other. The number of different APs between two collections increases with their time interval gradually. However, there is a large gap between AP components in the first collection and that in the last one, especially in a long time period. The gradual variations of common and semi-common APs help to bridge the gap.

2.1.2 Impact on Localization Accuracy. To examine how the number of common APs impacts on localization accuracy, we simulate decreasing number of APs by using Wi-Fi data collected at the first time. In order to avoid

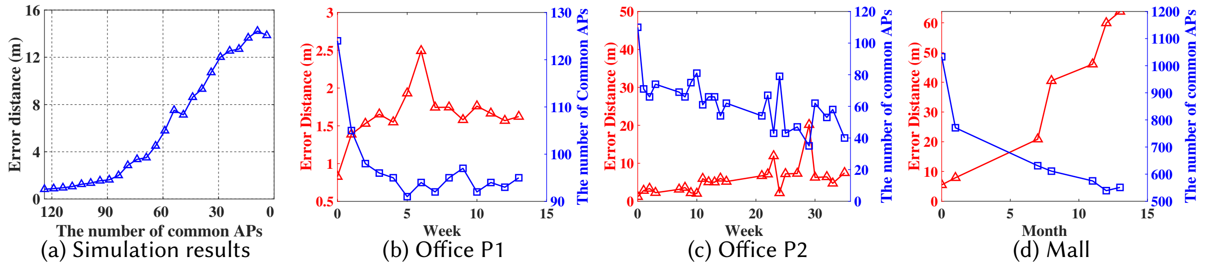


Fig. 2. The number of common APs impacts on the localization accuracy.

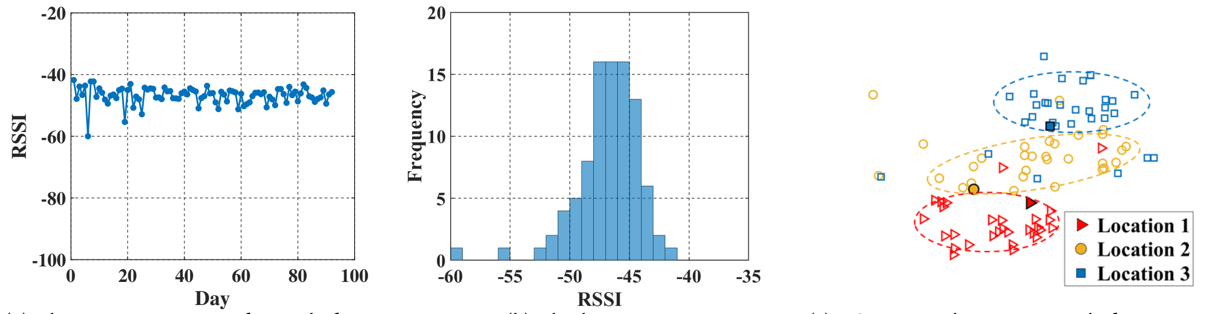
the impact of variation of Wi-Fi signals over time, both fingerprint database and test data are from Wi-Fi data in the first collection. The fingerprint database keeps fixed during the simulation. As for test data, we decrease the number of APs gradually. In particular, we choose $n(n \in [0, N])$ APs randomly from the total of N APs in the collection. These n APs are used to simulate common APs. Others simulate disappeared ones. For each run, we process the test data, keeping Wi-Fi signals from common APs and dropping the others. Then error distance is estimated by a plain K-nearest neighbors (KNN) algorithm. For each n, we run the simulation for 30 times to smooth the randomness, where n decreases from $N = 124$ to 4 with a step of 5.

The simulation results are shown in Fig. 2a. It illustrates that the error distance increases as the number of common APs decreases. Specifically, when the number of common APs is more than 90, the error distance increases slowly. Thereafter it increases fast. That is because the localization accuracy depends on the density of APs, i.e., sparse density of APs usually leads to poor accuracy [20].

Furthermore, on dataset of P1, P2, and mall we calculate error distances respectively, using Wi-Fi data collected at test points in each collection, keeping fingerprints as the first one. The results are shown in Fig. 2b, 2c, and 2d. We plot the error distance as well as the number of common APs for reference. We can see that as time goes by, the error distance increases with the decreasing number of common APs. On P1, from week 5 the number of common APs fluctuates up and down, so does the error distance. Note that there is a sharp increase at week 6. Examining the data, we found that two common APs are turned off during the week. When they are turned on again later, the error distance decreases back. Similarly, on P2 the number of common APs decreases to less than 40 at week 29, leading to a sharp increase of error distance (20 m). The same trends hold on dataset Mall.

2.1.3 Long-Term Characteristics of Wi-Fi Signals. To study the long-term variation of Wi-Fi signals, we analyze the RSSIs from common APs over time at a location (a reference point). Examining lots of signals from common APs at all reference points, we found that most of them have strong and stable signals over a long time at many reference points. Their histogram is similar to a normal distribution. We select one of them, and illustrate its time-sequence signals and histogram in Fig. 3a and 3b. The signals fluctuate up and down randomly with time. As for the histogram, out of 92 RSSIs only two points (−60 and −55) deviate from the center (−48), and others look like a normal distribution.

To further show the stability of Wi-Fi signals, we visualize the signals of common APs over a long time by t-SNE [33] at three adjacent reference points in Fig. 3c. A dot in the figure represents a signal vector composed of RSSIs from common APs at a reference point. The color of a dot represents its location. The filled dots are from the first collection, the others from subsequent collections. As illustrated in Fig. 3c, although a few of dots scatter near dots with a different color due to time variations of signals, most of dots are clustered according to their locations. The result shows that Wi-Fi signals from common APs are stable over a long time and thus have location discrimination to some extent.



(a) The time sequence of signals from a common AP

(b) The histogram

(c) t-SNE visualization: signals from common APs over time at three adjacent locations

Fig. 3. Wi-Fi signals from common APs in the long term.

Our experimental findings are summarized as:

- The number of common APs decreases over time, leading to poor localization accuracy.
- Two Wi-Fi collections with a large time interval share smaller number of APs than contiguous ones. Common APs and semi-common APs bridge the gap between the two collections far away from each other.
- Wi-Fi signals of most common APs are strong at many reference points and are stable over a long time.

2.2 Motivations

Our over-one-year experiments help us to understand long-term variations of APs and characteristics of their Wi-Fi signals. Since signals of common APs are stable over time, Wi-Fi signals sampled at two different times T_1 and T_2 have some features in common if they are at the same location. Therefore, the location information of signal at T_1 can be learnt and transferred to signal at T_2. Motivated by this, we apply unsupervised domain adaptation (UDA) to solve the fingerprint update problem.

In particular, taking collection time as a domain, the fingerprint database (labeled) is the source domain, and the collection of Wi-Fi signals sampled on line without location information (unlabeled) belong to target domains. Through domain adaptation, the location information can be learnt from the source domain and transferred to the target domain. As a result, the unlabeled Wi-Fi signals have location informations too.

Since there is a large gap between two Wi-Fi collections apart from each other, it is non-trivial to transfer label knowledge between them directly. Fortunately, many common and semi-common APs are shared among contiguous collections and vary gradually over time. Inspired by this, we adopt multiple targets in time sequence rather than one target, using common and semi-common APs to fill in the gap. Hence, we design a deep multi-target domain adaptation model MTDAN to learn common features among collections and to transfer location knowledge little by little.

2.3 Challenges

However, we have to address the following challenges:

- How to learn time-invariant and location-aware features so as to transfer label knowledge to unlabeled fingerprints? Although it is a primary function of domain adaptation, it is challenging for Wi-Fi fingerprinting. On one hand, since the number of common APs decreases with time, the shared information between the source and the target decreases too, which makes it difficult to align the two domains. Although multiple targets can help bridge the gap, aligning multiple target domains with the source poses a new challenge. On the other hand, despite the source and targets are aligned, minor disturbances of time-invariant features may have a significant impact on the prediction of locations. Therefore, it is challenging to design a perfect model for Wi-Fi fingerprinting system.

- How to save storage and learn time-specific features as well? Since the number of common APs decreases with time, we have to learn time-specific features from semi-common APs in subsequent collections. One straightforward way is to take each collection as a target domain, and train the model with all the targets when updating model. However, the storage for saving subsequent fingerprints and the training cost will grow explosively with time. Therefore, it is challenging to balance storage and learning of time-specific features.
- How to detect outliers in the fingerprints and avoid learning abnormal and temporary features from them? Outliers of Wi-Fi signals are usually caused by random events, such as turning on or off APs, body blockage, and temporary movements of objects. These events will change features of fingerprints but for a short time (usually limited in the current collection). If temporary features of an outlier are learnt, the subsequent location queries will be mis-located. However, if not learnt, current query will get a location with a high error. Moreover, it is challenging to detect outliers due to their randomness.

3 OVERVIEW OF MTLOC

Our goal is to design a practical Wi-Fi fingerprinting system maintaining high localization accuracy in the long term yet at a low cost, without the need of inertial sensor assistance or floor plan constraints. To this end, we propose MTLoc based on deep multi-target domain adaptation and overview it in this section.

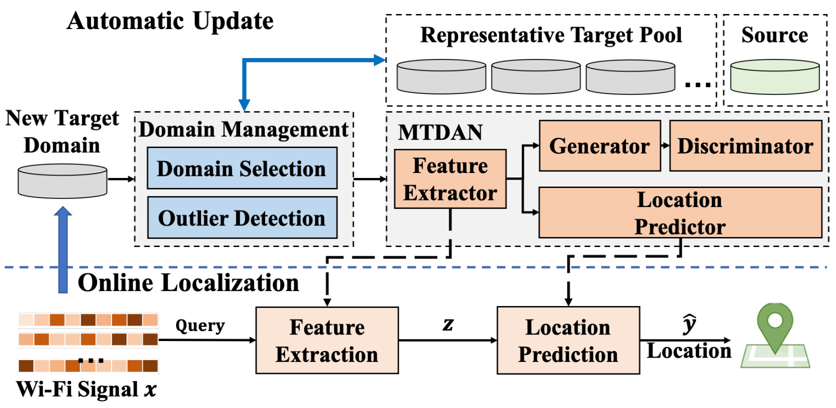


Fig. 4. The framework of MTLoc.

As illustrated in Fig. 4, MTLoc consists of two phases: automatic update and on-line localization. Generally, MTDAN model is updated automatically by training, then the well-trained model is used for on-line localization. In the phase of automatic update, new target domains are sequentially input to the domain management module which identifies them as representative, common, or outlier targets. A representative is stored in the representative target pool, whereas a common target is dropped directly. A new-detected outlier or a new representative triggers the update of MTDAN when MTDAN is trained by representatives in the pool (including the new one), the outlier (if any), and the source domain. Since we focus on the fingerprint update, we assume there has been a source domain (the initial fingerprint database) in advance. The target domains can be constructed during localization.

Specifically, Wi-Fi signals for location query are buffered continuously for a short period forming a new target domain. As time goes by, new target domains are constructed one by one.

In the phase of online localization, when a user queries his location, Wi-Fi RSSI vector x received by his her phone is first input to the feature extraction module (well-trained feature extractor) to extract its feature z. Then the location prediction module (well-trained location predictor) predicts the location and feeds it back to the user.

4 DESIGN OF MTDAN

MTDAN aims to extract time-invariant, time-specific, and location-aware features from the source and multiple target domains, transferring location knowledge from the source to targets. We now present the design of MTDAN in detail.

Let $x = (rssi_1, rssi_2, \ldots, rssi_M)$ denote a Wi-Fi fingerprint, where $rssi_i$ is the received signal strength indicator (RSSI) received from AP i, M is the total number of APs heard in an area of interest (AoI). A location in the AoI is represented as $y \in \mathbb{R}^2$. We consider one source domain and multiple target domains. Each fingerprint x_s in the source domain s is labeled and associated with a location y_s. Whereas, fingerprints in target domains are unlabeled. Assume there are N target domains in time sequence, denoted as target domain $t_1, t_2, \ldots, t_N$, respectively. Accordingly, x_{t_i} is a fingerprint in target domain $t_i, i \in [1, N]$. For the sake of convenience, we use a set $DI_{st} = \{s, t_1, t_2, \ldots, t_N\}$ to represent the index set of all domains.

As illustrated in Fig. 5, MTDAN consists of a feature extractor, a generator, a discriminator, and a location predictor. The feature extractor cooperates with other components to learn features from source and target domains. Particularly, by collaborating with the generator, the feature extractor learns time-specific features. Combining with the generator and discriminator, the feature extractor learns time-invariant features. Together with the location predictor, location-aware features are learnt as well. In addition, to enhance the alignment of source and target domains, we also design cycle consistency constraints on two levels: feature level and prediction level.

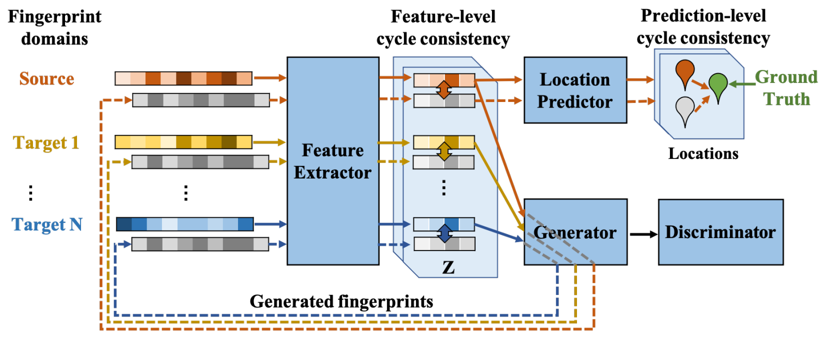


Fig. 5. The architecture of MTDAN.

4.1 Feature Extractor and Generator

In order to learn time-specific features of domains, the feature extractor and generator constitute a conditional autoencoder [31]. A fingerprint x_d in domain d ($d \in DI_{st}$) is firstly encoded to a latent space $\mathbb{Z}$ by the feature extractor, and then reconstructed to $\hat{x}_d$ back in domain d by the generator. The process is presented as $\hat{x}_d = G(E(x_d), d)$, where $E(\cdot)$ and $G(\cdot)$ are functions of the encoder and the generator, respectively. Let z_d be the feature of x_d in the latent space $\mathbb{Z}$, then we have $z_d = E(x_d)$. Therefore, the reconstruction process of fingerprint x_d can be written as

$$\hat{x}_d = G(E(x_d), d) = G(z_d, d). \tag{1}$$

To ensure $\hat{x}_d$ is as close as possible with x_d, the reconstruction loss $\mathcal{L}_{GE}$ is represented as:

$$\mathcal{L}_{GE} = \mathbb{E}_{x_d \sim P_d} \left[||x_d - \hat{x}_d|| \right], \tag{2}$$

where P_d is the distribution of domains joining the training including the source domain and the targets.

By encoding a fingerprint from a domain and remapping it back to the same domain, the feature extractor learns the time-specific (domain-specific) features.

4.2 Feature Extractor, Generator, and Discriminator

To learn time-invariant features, the feature extractor cooperates with the generator and discriminator as a conditional GAN [26]. Unlike the traditional GAN [6] whose fake samples are generated from random variables, we cascade the feature extractor to the generator for robustness [31].

For a fingerprint x_d in domain d, the feature extractor firstly encodes it to the latent space obtaining its feature z_d. On condition of domain $d', d' \in DI_{st}$, the generator generates its fake fingerprints $\hat{x}_{d \to d'}$ from z_d. The process can be expressed as

$$\hat{x}_{d \to d'} = G(z_d, d'). \tag{3}$$

In this way, a fingerprint in source domain x_s can be transformed to any target domain t_i, ($i \in [1, N]$), and vice versa. Furthermore, a fingerprint in a target domain can be transformed to any one of other target domains. That is

$$\begin{aligned} \hat{x}_{s \to t_i} &= G(z_s, t_i), \\ \hat{x}_{t_i \to s} &= G(z_{t_i}, s), \\ \hat{x}_{t_i \to t_j} &= G(z_{t_i}, t_j), i, j \in [1, N], i \neq j. \end{aligned} \tag{4}$$

And then, the real fingerprints and fake ones are input to the discriminator. By playing a minimax game, the discriminator tries to distinguish real fingerprints from fake ones, while the feature extractor and generator try to fool the discriminator by generating indistinguishable fake fingerprints.

Hence, the loss function is written as

$$\begin{aligned} \mathcal{L}_D = &\mathbb{E}_{x_s \sim P_s} \left[(D(x_s) - 1)^2 \right] + \mathbb{E}_{x_{t_i} \sim P_T} \left[(D(\hat{x}_{t_i \to s}))^2 \right] \\ &+ \mathbb{E}_{x_{t_i} \sim P_T} \left[(D(x_{t_i}) - 1)^2 \right] + \mathbb{E}_{x_s \sim P_s} \left[(D(\hat{x}_{s \to t_i}))^2 \right] + \mathbb{E}_{x_{t_j} \sim P_T} \left[(D(\hat{x}_{t_j \to t_i}))^2 \right], i, j \in [1, N], i \neq j. \end{aligned} \tag{5}$$

where $D(\cdot)$ is the function of the discriminator.

Through adversarial training, the feature extractor is encouraged to align the source and target domains in the latent space learning time-invariant (domain-invariant) features.

4.3 Feature Extractor and Location Predictor

To learn location-aware features, the feature extractor cascades a location predictor which predicts locations for Wi-Fi fingerprints. Instead of a classifier in [38], we adopt a regressor as the predictor due to its continuous output thus achieving high localization accuracy. Let $R(\cdot)$ denote the function of the regressor. For a fingerprint x_d in domain d, its location $\hat{y}_d$ is predicted by

$$\hat{y}_d = R(E(x_d)) = R(z_d). \tag{6}$$

Since only fingerprints in the source domain are labeled, we write the loss function of predictor $\mathcal{L}_R$ as

$$\mathcal{L}_R = \mathbb{E}_{x_s \sim P_s} \left[||\hat{y}_s - y_s|| \right]. \tag{7}$$

By minimizing the loss function, the predictor learns location knowledge from the source domain. At the same time, through adversarial training the feature extractor is driven to learn location-aware features. When the source and target domains are well aligned in the feature space, the location knowledge can be transferred to the target domains by (6).

4.4 Cycle-Consistency Loss

Even though the feature extractor is able to learn time-specific, time-invariant, and location-aware features by cooperating with other components, it is hard to accurately align the source and target domains in the feature space. This is because only fingerprints in source domain are associated with locations thus lacking pairwise alignment information between the source and the target.

To enhance the alignment, we introduce cycle-consistency constraints at two levels: feature-level and prediction-level [11]. The basic idea is that when a fingerprint in source domain is transformed to a target domain, the transformed fingerprint and the original one should be close in the feature space. Their locations predicted also should be close, and both are close to the ground truth.

4.4.1 Feature-Level Cycle-Consistency. Let x_s be a fingerprint in the source domain with label y_s. Its transformation in target domain t_i is $\hat{x}_{s\to t_i} = G(E(x_s), t_i)$. Then, in the feature space z_s and $z_{s\to t_i}$ should be close. Similarly, when a fingerprint in a target domain x_{t_i} is transformed to the source domain, the transformed fingerprint $\hat{x}_{t_i\to s}$ should be close to the original in the feature space. Let z_{t_i} and $z_{t_i\to s}$ be their features, then we denote the feature-level cycle-consistency loss function $\mathcal{L}_{CC}^{f}$ as

$$\mathcal{L}_{CC}^{f} = \mathbb{E}_{x_s\sim P_s}\left[||z_s - z_{s\to t_i}||\right] + \mathbb{E}_{x_{t_i}\sim P_T}\left[||z_{t_i} - z_{t_i\to s}||\right]. \tag{8}$$

By minimizing loss function $\mathcal{L}_{CC}^{f}$, the source domain and target domains are driven to be aligned in the feature space, enhancing the learning of time-invariant features.

4.4.2 Prediction-Level Cycle-Consistency. Despite $\mathcal{L}_{CC}^{f}$ encouraging the alignment, minor disturbances of features may still have a significant impact on the prediction of locations. Since only fingerprints in source domains are labeled, to further enhance the alignment we define prediction-level cycle-consistency loss $\mathcal{L}_{CC}^{p}$ as

$$\mathcal{L}_{CC}^{p} = \mathbb{E}_{x_s\sim P_s, t_i}\left[||\hat{y}_{s\to t_i} - y_s||\right], \tag{9}$$

where $\hat{y}_{s\to t_i} = R(E(x_{s\to t_i})) = R(z_{s\to t_i})$.

By minimizing loss function $\mathcal{L}_{CC}^{p}$, the predicted location of $x_{s\to t_i}$ ($\hat{y}_{s\to t_i}$) is as close as possible to the ground truth y_s which is the label of the original fingerprint x_s. Together with (7) which makes the predicted location of fingerprint x_s ($\hat{y}_s$) as close as possible to y_s, the predicted locations of $x_{s\to t_i}$ and its origin x_s are close too. Through adversarial training, the feature extractor is further driven to align the source and target domains in the feature space.

Putting it together, the total cycle-consistency loss $\mathcal{L}_{CC}$ is:

$$\mathcal{L}_{CC} = \mathcal{L}_{CC}^{f} + \mathcal{L}_{CC}^{p}. \tag{10}$$

Two-level cycle-consistency loss functions constrain the mapping of the source and target domains at both the feature level and the prediction level, enhancing the alignment in the feature space.

4.5 Objective

In summary, The objective of MTDAN $\mathcal{L}$ is written as:

$$\mathcal{L} = \mathcal{L}_{GE} + \lambda_D \mathcal{L}_D + \lambda_R \mathcal{L}_R + \lambda_{CC} \mathcal{L}_{CC}, \tag{11}$$

where λ_D, λ_R, and λ_{CC} are hyper-parameters to trade off different parts of the loss function.

5 DOMAIN MANAGEMENT

5.1 Domain Selection

To retain high localization accuracy, MTDAN has to be updated in time using new target domains. A new target domain is composed of up-to-date fingerprints collected at a low cost (e.g., by crowdsourcing, or by aggregating Wi-Fi signals on query in a short period). On one hand, if all the new targets join the update of MTDAN, the costs of storage and training will increase explosively with time. On the other hand, if only current target joins the update, the localization accuracy will decrease significantly with time due to the big gap between the source domain and the target. To balance the cost and accuracy, we select some of target domains (representative targets, or representatives in short) to store and drop others. Since target domains are collected gradually, they are input to MTDAN one by one. In this case, two questions arise: 1) how do we select representative targets? and 2) how can we represent the dropped ones?

Recall that contiguous targets share much more common and semi-common APs than those far away from each other. Therefore, we believe that contiguous targets cluster together. A representative target is then the one with the minimum divergence to others in the cluster. To choose the representative target, we first propose a measurement to measure the divergence between two target domains, and then design an online domain selection algorithm.

5.1.1 Domain Divergence Measurement. We define domain divergence between two domains as the expected difference between their fingerprints. For domain d and d', their domain divergence $\mathcal{D}(d, d')$ is defined as

$$\mathcal{D}(d, d') := \mathbb{E}_{(x_d, x_{d'}) \sim P_{d,d'}} \left[\| x_d - x_{d'} \| \right], \tag{12}$$

where $P_{d,d'}$ is the distribution of domain d and d'.

Since there is not pairwise information between two target domains, it is impossible to calculate $\mathcal{D}(d, d')$ by (12). To do so, we take the source domain as a benchmark. In particular, we rewrite $\|x_d - x_{d'}\|$ as $\|(x_d - x_s) - (x_{d'} - x_s)\|$, and approximate it as

$$\begin{aligned} \mathbb{E}_{(x_d, x_{d'}) \sim P_{d,d'}} \left[\| x_d - x_{d'} \| \right] &= \mathbb{E}_{(x_d, x_{d'}) \sim P_{d,d'}} \left[\| (x_d - x_s) - (x_{d'} - x_s) \| \right] \\ &\approx \| \mathbb{E}_{x_d \sim P_d} (x_d - x_s) - \mathbb{E}_{x_{d'} \sim P_{d'}} (x_{d'} - x_s) \| \\ &\approx \| \mathbb{E}_{x_d \sim P_d} (x_d - \hat{x}_{d \to s}) - \mathbb{E}_{x_{d'} \sim P_{d'}} (x_{d'} - \hat{x}_{d' \to s}) \|, \end{aligned} \tag{13}$$

where P_d, $P_{d'}$ are distributions of domain d and d', respectively. Furthermore, let

$$v_d = \mathbb{E}_{x_d \sim P_d} (x_d - \hat{x}_{d \to s}). \tag{14}$$

Now v_d can be calculated, since $\hat{x}_{d \to s}$ is transformed from x_d. Actually, v_d measures the distance between fingerprints in domain d and the corresponding one transformed to the source domain. Hereafter, v_d is named by *domain vector* of domain d.

Combining (12), (13), and (14), we can calculate the domain divergence $\mathcal{D}(d, d')$ as

$$\mathcal{D}(d, d') = \| v_d - v_{d'} \|. \tag{15}$$

In practice, to preserve more details, we replace the difference in (14) with concatenated positive and negative differences. Hence, v_d is calculated as:

$$v_d = \mathbb{E}_{x_d \sim P_d}[\text{ReLU}(x_d - \hat{x}_{d\to s}) \oplus \text{ReLU}(\hat{x}_{d\to s} - x_d)], \tag{16}$$

where $\oplus$ is the concatenation operator, and ReLU is a function to preserve non-negative values, written as

$$\text{ReLU}(a) = \begin{cases} a, & a > 0, \\ 0, & a \le 0. \end{cases} \tag{17}$$

In short, we can compute the domain divergence between any two domains by (15) even for the dropped ones, since it depends only on domain vectors instead of their fingerprints. What we need to do is to calculate the domain vector and record it when a new target domain comes.

5.1.2 Online Domain Selection Algorithm. Upon receiving a new target domain, if the representative target pool is not full, the new target directly becomes a representative. Otherwise, the online domain selection algorithm determines whether it is a representative or not. If so, the new one is stored in the pool; otherwise it is input to the outlier detection for later process.

To select a representative target, we propose a measurement, *the sum of squared domain divergence* (SSD), to indicate the overall divergence between each representative target and any one of other target domains. Suppose the representative target pool is able to store at most K targets. Let RT denote the set of representative targets. If we have processed $(n-1)$ target domains, the current SSD can be computed as

$$SSD = \sum_{t_i, i=1,\ldots,n-1} \min_j \{\mathcal{D}^2(t_i, j) | \text{for all representative target } j \in RT\}. \tag{18}$$

For a new target t_n ($n \in [1, N]$), we try to replace each representative target in the pool with the new one, and compute the SSD. When representative target k is taken place by the new target t_n, the set of representative targets becomes RT_k, $RT_k = RT - \{\text{target } k\} \cup \{\text{target } t_n\}$. SSD_k is calculated as

$$SSD_k = \sum_{t_i, i=1,\ldots,n-1} \min_j \{\mathcal{D}^2(t_i, j) | \text{for all representative target } j \in RT_k\}. \tag{19}$$

The SSD of target t_n, SSD_{t_n} is the minimum among all $SSD_k, k = 1, \ldots, K$, that is

$$SSD_{t_n} = \min_k \{SSD_k | \text{for all } k \in [1, K]\}. \tag{20}$$

We then compare SSD_{t_n} with current SSD. If SSD_{t_n} is smaller than SSD, then target t_n is chosen to be a representative one. In this way, the domain selection algorithm is able to select a representative target with the smallest SSD.

By domain selection, only representative targets are stored and used to train MTDAN. Therefore, the cost of storage and training is reduced significantly.

5.2 Outlier Detection

When a new target domain is not selected as a representative, the outlier detection module will examine it to see if it is an outlier. An outlier usually contains abnormal fingerprints caused by random events such as body blockage, APs turning off temporally, impermanent decoration, or object movement, especially in large-scale indoor environments (e.g., shopping mall). Limited by transmission range of Wi-Fi signals, the unexpected changes in fingerprints usually happen in one or several small areas and is not easy to detect them out.

In addition, these changes are often transient without any impact on subsequent target domains. If their features are learnt and remembered, the localization accuracy of queries in subsequent periods will be worsen. However, ignoring the outlier will make the current queries suffer from poor accuracy. Therefore, when an outlier is detected, how to still keep high localization accuracy against random and transient changes in fingerprints is the other challenge.

To address the first challenge, we design an outlier detection algorithm by leveraging cycle-consistency constraints. The basic idea comes from the deduction of the constraints: if a new target domain is close to the representative ones, its cycle-consistency loss is small. In other words, a significant increase in cycle-consistency loss indicates that the new target domain deviates from the representatives, i.e., it is an outlier. Since it is difficult to set a threshold on cycle-consistency loss, we turn to compare the loss of the current target with that of the previous. If the loss of the current target is more than most of that of the previous, it implies an outlier.

Specifically, we present an outlier indicator Q,

$$Q = \frac{1}{2}(Q^f + Q^p), \tag{21}$$

where Q^f and Q^p are feature-level and prediction-level outlier indicator, respectively.

5.2.1 Feature-Level Outlier Indicator. For a new target domain t_n, its feature-level cycle-consistency loss $CC^f_{t_n}$ is

$$CC^f_{t_n} = \frac{1}{K} \sum_{\text{domain } k \in RT} \|z_{t_n} - z_{t_n \to k}\|, \tag{22}$$

where $z_{t_n} = E(x_{t_n})$, $z_{t_n \to k} = E(\hat{x}_{t_n \to k}) = E(G(z_{t_n}, k))$.

Hence, the feature-level indicator Q^f is written as:

$$Q^f = \frac{1}{n-1} \sum_{t_i, i=1,\ldots,n-1} 1_{CC^f_{t_i} < CC^f_{t_n}}, \tag{23}$$

where 1_A is an indicator function of event A,

$$1_A = \begin{cases} 1, & \text{if A happens,} \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

A larger Q^f indicates a higher probability that target t_n is an outlier.

5.2.2 Prediction-Level Outlier Indicator. Similarly, the prediction-level cycle-consistency loss of target t_n, CC^p_d, is represented as:

$$CC^p_{t_n} = \frac{1}{K} \sum_{\text{domain } k \in RT} \|\hat{y}_{t_n} - \hat{y}_{t_n \to k}\|, \tag{25}$$

where, $\hat{y}_{t_n} = P(z_{t_n})$, $\hat{y}_{t_n \to k} = P(z_{t_n \to k})$.

Hence, Q^p is written as:

$$Q^p = \frac{1}{n-1} \sum_{t_i, i=1,\ldots,n-1} 1_{CC^p_{t_i} < CC^p_{t_n}}. \tag{26}$$

A larger Q^p indicates a higher probability that target t_n is an outlier.

Therefore, a large value of outlier indicator Q indicates a higher probability that the new target is an outlier. Furthermore, we set a threshold δ_Q on the outlier indicator. if the value of its Q is larger than the threshold δ_Q, the target domain is detected to be an outlier.

To tackle the second challenge, we make use of *forgetting* in deep learning [21] which says that a model will forget what has been learnt previously in a sequence learning. Specifically, when an outlier is detected, it triggers the update of MTDAN and joins the training. After the update, the outlier is dropped. The only once of training allows MTDAN to learn the transient features in current period and then forget them later. In this way, MTDAN is able to handle random changes in fingerprints and limit their impact on the subsequent localization.

6 TRAINING STRATEGIES

MTDAN is updated by training. To improve the efficiency of training, we design four training strategies: adaptive update, limited volume of training data, alternating complete training and fine-tuning, and pre-training.

Adaptive Update. Instead of periodic update, we design a dynamic update adapting to the changes of fingerprints. The update is triggered in two cases: 1) when a new target domain is selected as a representative, MTDAN is updated to learn new features with permanent changes in fingerprints; and 2) when the new target is an outlier, MTDAN is updated to learn temporary features handling transient changes. Whereas, if the new target is common with little changes, the model will not be updated. In this way, MTDAN is updated in time to keep high localization accuracy and the number of updates is reduced as well.

Limited Volume of Training Data. Since training data only consist of the source domain, the outlier (if any), and representative targets in the pool, the volume is limited thus avoiding explosive growth with time. As a result, the training time is limited too.

Alternating Complete Training and Fine-Tuning. Recall that Wi-Fi fingerprints in one AoI usually change gradually with little difference among contiguous domains. Therefore, we alternate complete training and fine-tuning to improve training efficiency.

As a complete training is performed from scratch on all training data (the source domain, the outlier (if any), and representative target domains in the pool), it takes a long time and yet is able to be trained well. Generally, complete training is carried out at the beginning and during the long run. Whereas, fine-tuning refers to a training based on parameters obtained in the last training. Compared with complete training, fine-tuning can converge with a less number of epochs thus reducing training time. However, since it depends on last training, fine-tuning has to follow a complete training or another fine-tuning. In addition, its capability of learning is constrained. Therefore, we perform a complete training at the beginning followed by several fine-tunings, and then a complete training, and so on alternately. In this way, MTDAN can be trained well at a low cost.

Pre-Training. There are dependencies among the components of MTDAN. To generate undistinguishable fake fingerprints, the generator relies on the meaningful representation of the feature extractor. Similarly, the location predictor also relies on the feature extractor for high accurate estimation. Therefore, it is crucial to provide proper initial parameters for joint training.

To this end, before joint training we pre-train each component of MTDAN for robustness and quick convergence. Specifically, we pre-train the feature extractor and generator with reconstruction loss $\mathcal{L}_{GE}$ to learn a meaningful representation of fingerprints. Fixing the feature extractor, the location predictor is then pre-trained with $\mathcal{L}_R$. After pre-training, the parameters of components are used in the joint training.

The joint training is in an adversarial pattern. Let $\theta_E, \theta_G, \theta_R$, and θ_D be the parameters of the feature extractor, generator, location predictor, and discriminator, respectively. With back-propagation of the gradients, θ_D is updated by the gradients of $\lambda_D \mathcal{L}_D$. Then, the gradients are multiplied by a negative constant η ($\eta < 0$) through a gradient reversal layer (GRL) [4] before propagating back to the generator. θ_E, θ_G and θ_R are updated with gradients of $\mathcal{L}_{GE} + \lambda_R \mathcal{L}_R + \lambda_{CC} \mathcal{L}_{CC}$ and gradients back-propagated from GRL.

7 IMPLEMENTATION AND EVALUATION

7.1 Implementation

MTLoc system is deployed at a server with CPU Intel Core Processor I7-7700K, 32G memory, and GPU GeForce GTX 1080Ti. We implement MTDAN with Pytorch 1.10.0 platform [28]. The sensing functionality of the system is implemented on Android smartphones.

7.1.1 Network Architecture. Fig. 6 illustrates the network architecture of MTDAN. We implement all components with Convolutional Neural Networks (CNNs). Each CNN block (ConvBlk) consists of a convolutional layer, batch

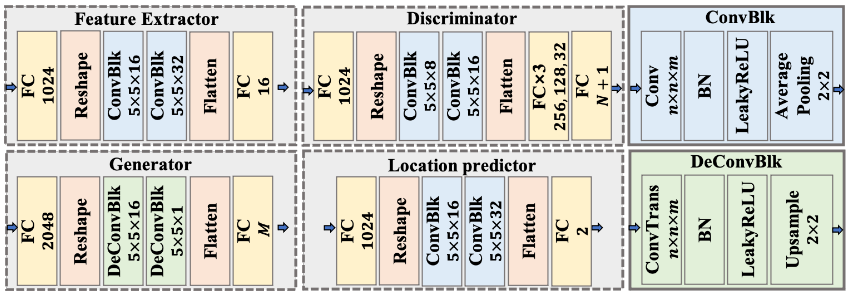


Fig. 6. Network architecture of MTDAN.

normalization (BN) [13], LeakyReLU [46], and an average pooling layer. Whereas, the Deconvolutional CNN block (DeConvBlk) has a network structure reverse to ConvBlk. The feature extractor consists of an FC layer, a reshape layer, two CNN blocks, a flatten layer, and a FC layer. Note that the Wi-Fi fingerprint is 1D and its size varies with the number of APs. To fit a 2D convolutional layer and handle the variation of the size, the feature extractor projects input fingerprints into a vector by an FC layer and then reshapes it to a matrix with a fixed size. Being reverse to the feature extractor, the generator involves two DeConvBlks. To stabilize the training of the GAN, the discriminator adopts spectral normalization [23]. The activation function in input FC layer and hidden FC layers use LeakyReLU. The output FC layer of the discriminator uses sigmoid function. Whereas, there are not activation functions in other FC layers.

7.1.2 Hyper-Parameter Setup. For all components of MTDAN, we use Adam [15] as the optimizer with a learning rate of 0.0002 and betas of (0.5, 0.999). MTDAN is pre-trained for 30 epochs and joint-trained for 100 epochs in complete training. For fine-tuning, MTDAN is joint-trained for 30 epochs. MTDAN performs a complete training every 10 fine-tunings. The batch size is 16 for the office, 64 for the mall, and 8 for public dataset of Lib-UJI [22]. We set K (the maximum number of representative targets) to be 5 for the office and Lib-UJI, and 2 for the mall. The threshold of outlier indicator δ_Q is set to 0.75 for the office and Lib-UJI, and 0.35 for the mall. λ_D, λ_R, λ_{CC} are set to 1. The negative constant η in gradient back propagating is set to -10 in all experiments.

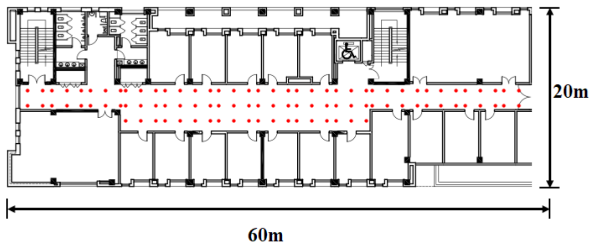
7.2 Experiment Setup

Table 1. Datasets. Size: size of the venue. Interval: interval between reference points (RPs); No. of APs: the number of ambient APs heard during collections. Note that we do not deploy any APs in the experiments.

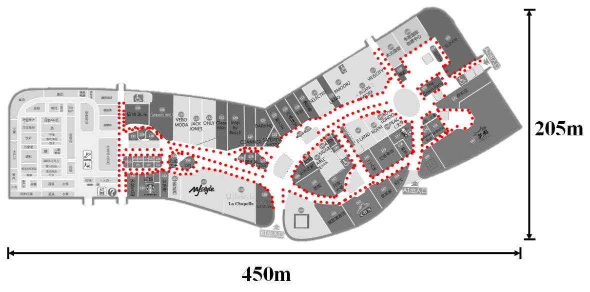
Datasets		Size (m^2)	Interval (m)	No. of RPs	No. of TPs	Training Samples	No. of Domains	Duration (Month)	No. of APs	Collection Frequency
	Office P1					62K	92	3	409	Daily
Ours	Office P2	200	1.5	97	92	16K	24	9	287	Weekly
	Office P1+P2					26K	38	15	361	Weekly
	Mall	8000	5	300	99	126K	7	13	2308	Monthly
UJI	Lib-UJI [22]	150	2	24	106	7K	25	25	280	Monthly

7.2.1 Datasets. To evaluate MTLoc, we use five datasets listed in Table 1. Four of them (Office P1, Office P2, Office P1+P2, and Mall) are collected by ourselves and publicly available in GitHub. Lib-UJI is released by the University Jaume I (UJI) [22]. Our datasets are from an office building and a shopping mall with a total area of

8, 200 m^2. The floorplans are shown in Fig. 7, where dots indicate locations of reference points (RPs). The test points (TPs) are disjoint with RPs. There are 97 RPs with 1.5 m intervals, and 92 TPs. In the shopping mall, there are 300 RPs with 5 m intervals and 99 TPs. Wi-Fi signals collected at RPs and TPs are used as training dataset and test dataset, respectively. Sampling rate is 5 Hz. Sampling time at each RP is 8 s in the office building, and 60 s in the shopping mall. Sampling time at each TP is 1 s in both scenarios.



(a) Office building



(b) Shopping mall

Fig. 7. Floor plans in experiment scenarios.

Due to holidays and epidemic of new coronavirus, our collection was interrupted from time to time. Office P1 and Office P2 are collected at different frequencies in two time periods. The former is collected daily in 3 months from September to December, 2020. The latter is collected weekly over 9 months from April to December, 2021. To obtain a long-term dataset, we combine Office P1 and P2 by downsampling P1 to once-a-week. As a result, Office P1+P2 covers 15 months. In the mall, we collected fingerprints monthly over 13 months. Due to the interruptions, there are only 7 domains (collections) in total. For all datasets, the first collection is used as the source domain, and the others are targets.

Dataset Lib-UJI [22] is collected in a library over 25 months, to the best of our knowledge, which is the open dataset covering the longest time period. For more details, please refer to Table 1.

Wireless environments and building layouts are very different in the above three indoor venues. While they are relatively stable in the office and library, both the wireless environments and building layouts change frequently in the shopping mall. During our data collection in the mall, many changes and events have happened, such as store movement along with their APs, store closing or new one opening, store decoration, and sales activities. All these events make the wireless environments highly dynamic in the mall. Moreover, the shopping mall has a larger area and more APs than others, which means more challenges in the long run (note that APs are those heard during collections, which are not deployed by us for experiments). Although the areas are similar in size in the office and library, they are different in space. It is a long corridor in the office, but it is wide open in the library leading to a different wireless environment. In short, the datasets are very typical in indoor environments.

7.2.2 Benchmarks. We take three state-of-the-art deep-learning-based models or approaches as benchmarks. Among them, DANN [4] is a UDA model; iToLoc [18] and CNNLoc [14] are localization approaches. Since the authors do not open their source codes, we have implemented them according to their papers.

DANN is a UDA model under an adversarial training framework, which is designed for image classification. However, our MTDAN adopts a regressor for localization since its continuous output can provide higher precision compared with a classifier. Therefore, we replace the classifier in DANN with a regressor for fairness. Moreover, the corresponding components use the same network architectures as that in MTDAN. Since DANN does not support automatic updates, when a new target comes we train it on both source domain and the new one.

iToLoc [18] is the newest localization system based on DANN and co-training. It learns dynamics-resistant and device-independent features of Wi-Fi RSS fingerprints by DANN, and updates itself automatically by co-training with unlabeled fingerprints.

CNNLoc [14] is a localization approach based on CNN. Since it does not consider model updates, we train CNNloc only on the source domain and take it as the baseline without updates.

7.3 Localization Accuracy

We evaluate the localization accuracy of MTLoc on both short-term (three months) and long-term datasets from various indoor environments. Office P1 is the relatively short-term dataset where target domains are collected every day. Since its wireless environment is simple and the building layout seldom changes, it is a good scenario to evaluate the basic design of MTLoc in detail. The long-term datasets are Office P2 (9 months), Office P1+P2 (15 months), Mall (13 months), and Lib-UJI (25 months). The interval of two contiguous targets varies from a week to one or several months. Through the long-term datasets, we examine the design of MTLoc in different environments, especially those complex, dynamic, and large-scale ones (e.g., shopping mall) in the long run.

Table 2. Summary of performance comparison. Metric is error distance in unit of meter. Office P1 (3 M): dataset Office P1 which covers 3 months (3 M). The other column titles have similar meanings. Init: the initial error distance at the beginning of the period covered by the dataset, where MTDAN is trained only by source domain. Final: the final error distance at the end of the period. Diff: the difference of error distances between final of current approach and that of MTLoc.

	Office P1 (3 M)			Office P2 (9 M)			Office P1+P2 (15 M)			Mall (13 M)			Lib-UJI (25 M)		
	init	final	diff	init	final	diff	init	final	diff	init	final	diff	init	final	diff
MTLoc	**0.5**	**1.0**	-	**0.8**	**1.2**	-	**0.6**	**1.2**	-	**4.5**	**7.8**	-	**2.7**	**2.3**	-
DANN	0.6	2.1	1.1	1.0	2.9	1.7	0.6	5.3	4.1	4.2	14.7	6.9	2.8	3.4	1.1
iToLoc	0.7	2.1	1.1	1.1	4.5	3.3	0.7	6.2	5.0	8.1	26.2	18.4	2.8	3.2	0.9
CNNLoc	1.1	2.7	1.7	1.6	7.0	5.8	1.1	9.3	8.1	13.1	22.1	14.3	3.0	3.5	1.2

Table 2 summarizes the results compared with that of three benchmarks (DANN, iToLoc, and CNNLoc). We can see that MTLoc outperforms the benchmarks on all the datasets. On Office P1, the final error distance of MTLoc after three months keeps 1.0 m, which is better than that of benchmarks by 1.1 m, 1.1 m, and 1.7 m, respectively. In the long term, on Office P1+P2, the finial error distance of MTLoc retains 1.2 m after 15 months, much better than that of the benchmarks by 4.1 m, 5.0 m, and 8.1 m, respectively. Even in more complex and dynamic mall, MTLoc still achieves 7.8 m after one year, outperforms the benchmarks by 6.9 m, 18.4 m, and 14.3 m, respectively. We analyze the results in depth from perspectives of environment changes and technique design as follows.

7.3.1 Three-Month Performance. We first evaluate the localization accuracy of MTLoc on a three-month dataset of Office P1. The results are illustrated in Fig. 8, where accuracy at day 0 refers to the initial localization accuracy on the source domain. We can see that the initial accuracy of MTLoc and the benchmarks are very high at sub-meter level except for CNNLoc. As time goes by, only MTLoc keeps the accuracy close to 1 meter, while others decrease significantly.

To understand the results, we plot the number of shared APs in Fig. 9, where the number of common APs quickly decreases to 97 on day 10, and then fluctuates between 90 and 100. On the other hand, the number of common APs and semi-common APs keeps a high level, even larger than the initial number (124) after 45 days. It means that many new APs appear with time and stay becoming semi-common APs.

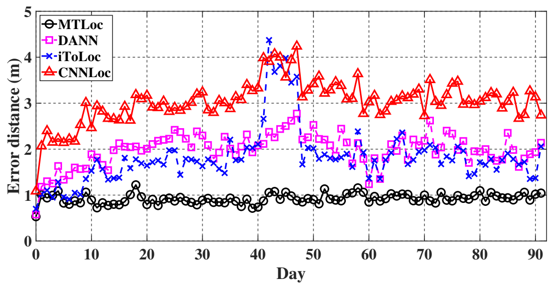


Fig. 8. Daily performance on Office P1.

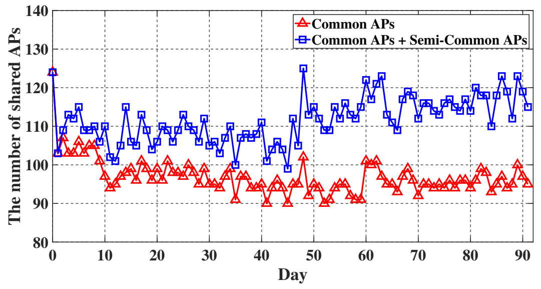


Fig. 9. The number of shared APs on Office P1.

The variation of the number of shared APs reveals why MTLoc retains high accuracy throughout three months. When the number of common APs decreases, there is a big gap in fingerprints between the source and the current target domain. MTLoc leverages semi-common APs to fill the gap between source domain and targets by training itself on multiple domains (source and representative targets). Besides, the constraints of cycle consistency enhances the alignment among source and targets. Therefore, the source domain and targets are aligned well leading to high accuracy. Whereas, DANN is trained only on source and current target thus suffering from poor accuracy when there is a big gap. Similarly, the accuracy of iToLoc also decreases and fluctuates significantly with time. The accuracy of CNNLoc is the lowest, since it does not update itself with time.

Zooming in on the error distance, there is a sharp increase from day 41 to 47 for the three benchmarks, whereas it is only a little fluctuation for MTLoc. As shown in Fig. 9, the number of common APs decreases to 90 on day 41 and 45, reaching the smallest value across the three months. The sharp decrease of the number of common APs causes a significant change in fingerprints. As a result, the error distances of iToLoc, DANN, and CNNLoc decrease sharply. However, MTLoc detects the change of fingerprints by outlier detection module where target domain on day 41 is detected as an outlier. The outlier triggers the update of MTDAN, through which changes of fingerprints are learnt quickly. In addition, the number of common and semi-common APs increases from day 45 to 48, meaning many new APs appear and stay. Consequently, targets on day 44 and 47 are selected to be representatives triggering updates of MTDAN. Therefore, MTLoc is able to learn fingerprint changes in time by outlier detection and representative selection.

7.3.2 Long-Term Performance. We then evaluate the long-term (9 to 25 months) performance of MTLoc in various scenarios, including office building, shopping mall, and library. Four datasets are involved, ours (Office P2, Office P1+P2, Mall) and the public dataset of Lib-UJI. The results are shown in Fig. 10. We can see that in all scenarios, MTLoc outperforms the benchmarks. Compared with the office and library, the mall experiences a more complex environment. During our data collection, many changes have happened. For example, one store moves to another place in the mall taking its AP along with it; one store moved out of the mall and a new one came in later; two stores were closed for a period for decorations; sometimes, the concourse was separated temporally for sales activities. All these events make fingerprints change significantly. Even in such a dynamic environment, after 13 months the error distance of MTLoc still reaches 7.8 m, outperforming its counterparts significantly.

MTLoc is able to keep high accuracy after a long time primarily due to its multi-target domain adaptation. After a long time, the number of common APs in each scenario decreases substantially by at least 50%. In particular, we plot the number of shared APs on Office P1+P2 and Lib-UJI in Fig. 11. By the end of the collection, the number of common APs decreases from 124 to 51 on Office P1+P2, and from 73 to 39 on Lib-UJI. In addition, because of a long interruption (over three months) of collections between P1 and P2, there is a sharp decrease at month 27 in Fig. 11a. There is also a sharp decrease at month 11 in Fig. 11b. The significant decrease in the number of common APs makes a big difference in fingerprints between the source and the target domain. DANN, iToLoc,

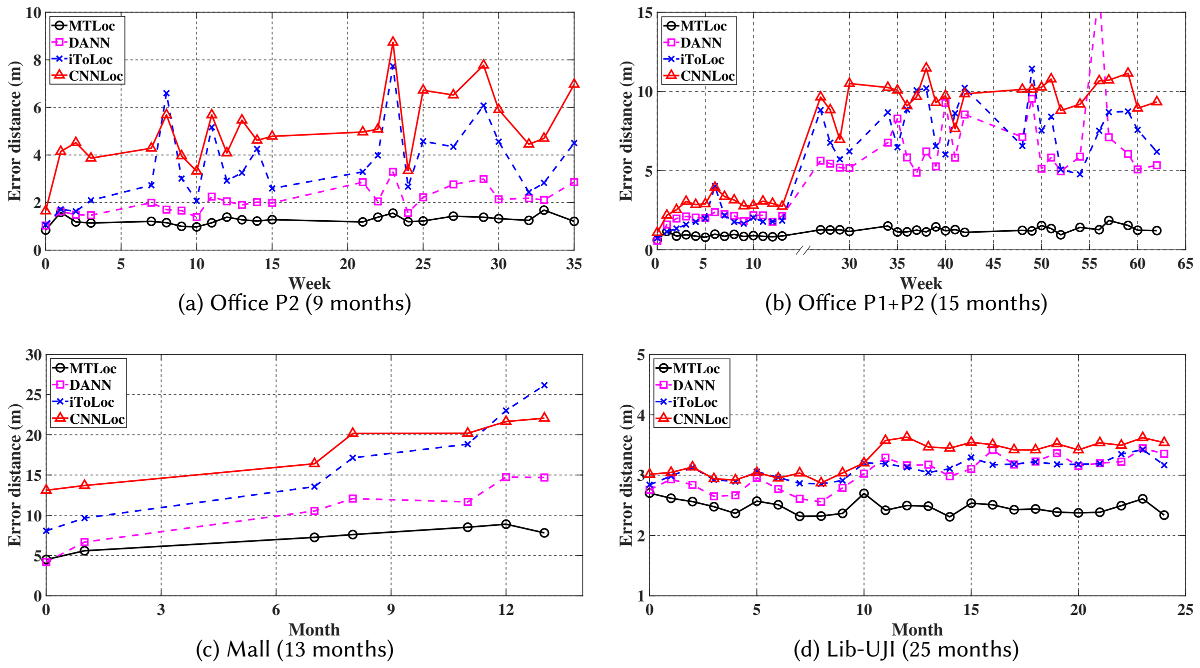


Fig. 10. Long-term Performance on various datasets.

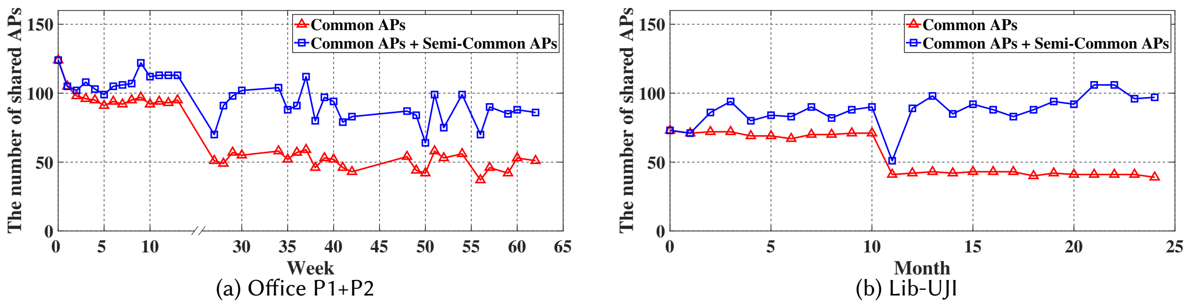


Fig. 11. The number of shared APs on datasets of Office P1+P2 and Lib-UJI.

and CNNLoc fail to learn the sharp changes of fingerprints thus suffering from poor performance. In contrast, MTLoc makes use of semi-common APs to bridge the source and the target by multi-target domain adaptation thus retaining high accuracy in the long run.

Besides multi-target learning, domain management module also contributes to the high accuracy of MTLoc. Outlier detection enables MTLoc to learn new transient features. Taking dataset Mall for example, at month 8 the target is detected as an outlier by MTLoc. At that time, a store was being decorated which causes changes of fingerprints around it. As the outlier triggers the update of MTDAN, the new features are learnt by MTLoc in time. Moreover, since outlier joins training only once and then is dropped, the transient features caused by temporary decoration have little impact on the targets after that. Therefore, the error distance of MTLoc increases a little bit at month 8 as shown in Fig. 10c, which is different from others.

In addition, MTLoc learns new long-term features that appear and stay later on through representative targets. Taking dataset Office P1+P2 for example, at week 27 a large number of common APs disappear, while from week 28 to 30 many new APs appear and stay (i.e., semi-common APs). The variation of APs makes big changes of fingerprints. MTLoc selects targets during this period as representatives. Since representatives join training each time for model update, MTLoc is able to learn new long-term features in time. Therefore, the error distance of MTLoc still keeps at a low level even after a long time.

In a nutshell, benefitting from multi-target domain adaptation and domain management, MTLoc achieves high accuracy in the long run, outperforming the benchmarks significantly on all datasets tested.

7.4 Effectiveness of Components in MTDAN

We carry out an ablation study to show the effectiveness of components in MTDAN. In particular, we take off one component at one time and keep others. The feature extractor, generator, and location predictor are preserved in each experiment since they are core components for localization. The parameters used are consistent with MTDAN. We denote approaches as follows:

- **noDiscriminator.** The discriminator is deleted to show the effectiveness of GAN framework.
- **noCycle.** Cycle consistency constraints are removed to show the effectiveness of the cycle consistency. Since outlier detection depends on the cycle consistency constraints, outlier detection is also removed in this approach.
- **singleTarget.** Domain management and representative target pool are removed to show the effectiveness of multi-target. Each time a new target comes, MTDAN is updated with the source and the new target domain (single target).

The experiments are performed on dataset Office P1+P2 and Mall. Fig. 12 shows the final error distance at the end of time period covered by each dataset. We can see that MTDAN outperforms all other approaches on both datasets.

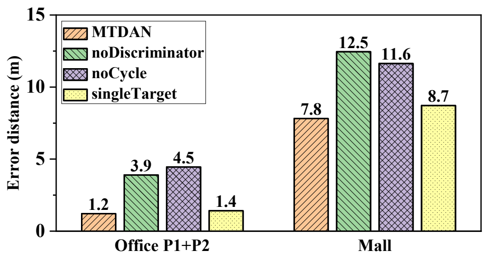


Fig. 12. Effectiveness of each component in MTDAN.

7.4.1 *Effectiveness of Discriminator.*

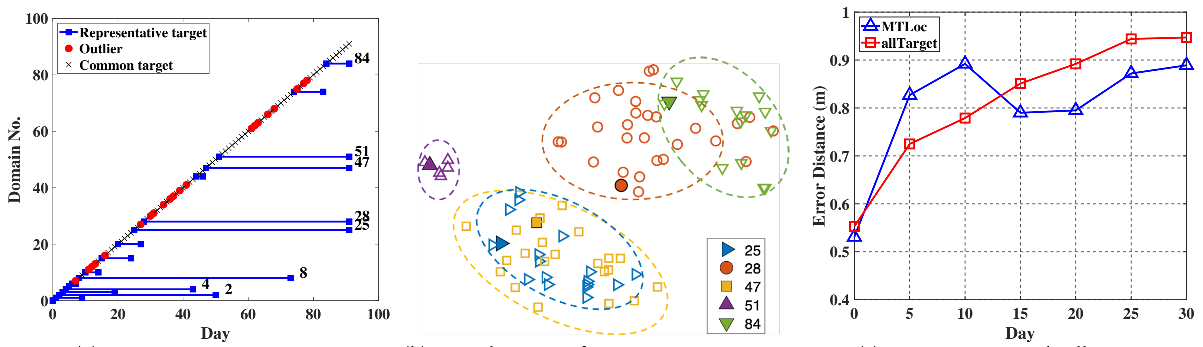
As shown in Fig. 12, without the discriminator the final error distances increase to 3.9 m and 12.5 m on Office P1+P2 and Mall, respectively. This is because that the discriminator and other components are under a GAN framework. By adversarial training, the feature extractor is driven to produce time-invariant and location-aware features. Therefore, the localization accuracy deteriorates when the discriminator is removed.

7.4.2 Effectiveness of Cycle-Consistency Constraints. Fig. 12 shows that without cycle-consistency constraints the final error distances increase to 4.5 m and 11.6 m on Office P1+P2 and Mall, respectively. The cycle-consistency constraints enhance the alignment among the source and targets in feature space. When they are removed, a small perturbation in the feature space may cause a big mismatch for location prediction. Therefore, the final error distance increases significantly.

7.4.3 Effectiveness of Multi-Target. Fig. 12 illustrates that the final error distances of singleTarget deteriorate considerably compared with that of MTLoc. On one hand, the number of common APs reduces with time especially after a long time. On the other hand, semi-common APs appear gradually. The variation of APs makes fingerprints change with time. With multi-target domain adaptation, MTLoc bridges the source and targets by leveraging intermediate representative targets. Therefore, MTLoc achieves high accuracy in the long run.

7.5 Validation of Domain Management

7.5.1 Validation of Domain Selection. Rather than using all targets, domain selection module selects representative targets to store and use them to train MTDAN. To validate the domain selection, two natural questions are: i) whether the selected representatives are able to represent all other targets? and ii) does it outperform the approach without domain selection (allTarget)? We then answer the questions by examining the results of MTLoc and allTarget on datasets. Due to space limitations, Fig. 13 only shows the results on dataset Office P1. The results hold on other datasets.



(a) Representative targets (b) Visualization of representative targets (c) Comparison with allTarget

Fig. 13. Domain selection on dataset Office P1.

i) *Selection records of representative targets.* We draw the records of representatives as well as outliers and common targets in Fig. 13a. The y-axis is the target domain number, and x-axis is the day number counted from the day when the source is collected. For the sake of convenience, we type the target number at the end of a line segment. For example, target 8 is selected as a representative on day 8 and stored in the representative target pool till day 75. We can see that since the maximum number of representatives K is set to 5, the first 5 targets are selected as representatives by default. After that, target 8, 10, and 15 are selected as representatives replacing one already in the pool. Referring to Fig. 9, the number of shared APs on day 8 is similar to that on day 7, which means some common APs disappear and some semi-common APs appear and stay. The corresponding changes of fingerprints in previous targets are involved in target 8, that is why MTLoc selects target 8 as a representative. The same goes for other representative selection. Therefore, a representative involves new features that appear in previous targets and stay thereafter, i.e., long-term features, which bridges source and targets in fingerprints.

In addition, throughout the whole experiment representatives in the pool are quite stable. Some of them keep a long time in it, such as target 8, 25, and 28. In this sense, representative selection is efficient. In short, the records demonstrate that representatives involve long-term features filling the gap between source and targets, and the domain selection is efficient.

ii) *Visualization of representatives and other targets.* According to the online domain selection algorithm, targets are clustered by their SSD measurement to a representative which has the smallest SSD. Therefore, a representative is the centroid of a cluster. To validate it, we consider the targets at the end of time period covered by the dataset, which include representatives 25, 28, 47, 51, and 84, and other targets. We concatenate all vectors of RSSIs in a target domain into one vector and visualize them by t-SNE. The results are shown in Fig. 13b. The targets with the same markers are those have the smallest SSD to the same representative. The filled markers denote the representatives. We can see that even though some clusters are overlapped, targets with same markers really cluster together. The results show that the representatives are able to represent all targets.

iii) *Comparison with allTarget.* The core of allTarget is MTDAN too. When a new target comes, allTarget uses all previous targets, the new one, and the source to update MTDAN without domain selection. Whereas, MTLoc uses only representatives and the source for updates. Fig. 13c illustrates their accuracy on dataset of P1. The accuracy of MTLoc is worse than that of allTarget at the beginning, but becomes better with time. Since the total number of representatives on P1 is set to 5, MTLoc and allTarget have the same dataset to train MTDAN before day 5. To save training cost, MTLoc completely trains MTDAN at day 0, and then keeps fine-tuning until day 10. That is why allTarget outperforms MTLoc during this period. MTLoc carries out a complete training at day 11 causing decrease of error distance. At the same time, with more and more targets joining the training, it is hard for allTarget to train MTDAN well. Therefore, MTLoc achieves better accuracy. The results demonstrate that MTLoc outperforms allTarget in the long run with limited number of representatives.

7.5.2 Validation of Outlier Detection. Outlier detection makes use of cycle-consistency constraints to detect fingerprint changes in targets. To validate it, we examine outliers detected on datasets. On Office P1, as shown in Fig. 13a targets on day 11, 12, and 13 are detected as outliers. During these days the number of semi-common APs varies significantly, first decreasing down to 105 and then increasing up to 121. The variation of APs results in big changes of fingerprints, and thus outliers are detected. After changes happen for three days, target 15 is selected as a representative involving the changes in its fingerprints. On dataset of Mall, the outlier is target 5 where fingerprints change temporally in some regions due to store decoration and recover in the next target domain. In a nutshell, the outliers involve fingerprint changes caused by variation of APs or by other random events (e.g., decoration, people flow in buildings). Therefore, outlier detection is effective.

7.6 The Cost of Storage and Training

By multi-target domain adaptation and domain management, MTLoc achieves high accuracy yet at a low cost in the long run. We now evaluate its cost in terms of the amount of storage and training time on various datasets, comparing it with allTarget approach. AllTarget uses MTDAN without domain selection. Each time a new target comes, it updates MTDAN with the new and all previous ones.

7.6.1 Storage Cost. Since MTLoc stores at most K representative targets in the pool, its storage cost is limited and does not increase with time. In contrast, allTarget has to store all the targets causing an increasing storage with time. Fig. 14 shows their storage cost at the end of time period covered by each dataset. The Y-axis uses log scale to illustrate the results clearly. Although the storage varies with datasets due to different sizes, collection frequency, and time periods, MTLoc requires much less storage than allTarget on all datasets. On P1, MTLoc stores 7.8 MB data saving 110.8 MB (by 93.4%) compared with allTarget. On Mall, MTLoc saves storage of 623.5

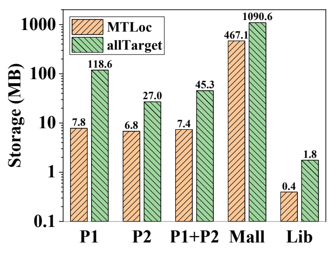


Fig. 14. The total storage.

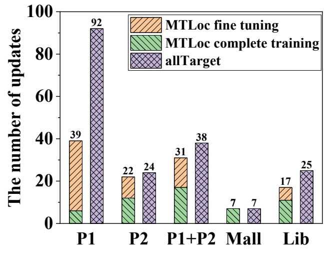


Fig. 15. The total number of updates.

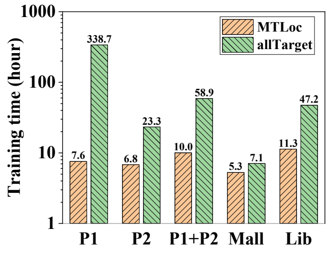


Fig. 16. The total training time.

MB (57.2%). On other datasets, the saving ratio reaches from 70% to 85%. As time goes on, MTLoc still keeps a fixed storage thus saving more and more storage. Therefore, MTLoc is at a low cost of storage.

7.6.2 Training Time. MTLoc decreases the training time from three aspects: reducing the number of updates, alternating complete training and fine-tuning, and limiting the number of target domains that join the training. MTLoc updates MTDAN only when a new target is a representative or an outlier. If the new one is common with little change of fingerprints, then MTLoc does not update. As a result, MTLoc reduces the total training time. Furthermore, for updates MTLoc alternates complete training and fine-tuning. Since fine-tuning trains MTLoc based on the parameters learnt in the last training, it converges quickly thus yielding a short training time.

Fig 15 illustrates the number of updates on various datasets. As the number of updates of MTLoc equals the sum of the number of complete training and that of fine-tuning, we stack them in a bar. We can see the number of updates of MTLoc is less than that of allTarget except on dataset of Mall. On Mall, there are only 6 targets in total over one year. Due to the large gap between contiguous targets, MTLoc trains MTDAN completely each time a new target comes. Therefore, MTLoc and allTarget have the same number of updates. On other datasets, MTLoc updates less than allTarget. Especially on P1, the number of complete training is only 6, while the number of fine-tuning is 33. The total number of updates of MTLoc decreases to 39 much less than that of allTarget by 57.6%.

Fig. 16 shows the total training time throughout a whole dataset. MTLoc has a smaller training time than allTarget on all the datasets. The total training time of allTarget is estimated according to its training time on a small number of targets. This is because when the number of targets becomes large, the training time of allTarget grows too long to be endured. On P1, the total training time of MTLoc is 7.6 hours, much shorter than that of allTarget (338.7 hours). This is primarily due to its small number of updates where fine-tuning takes 84.6%(= 33/39). The small volume of training dataset (less than allTarget by 92.4%) also makes a big contribution. On Mall, even though its number of updates is the same with that of allTarget, MTLoc has a smaller training time than allTarget due to its small volume of training dataset.

All in all, MTLoc keeps a low cost of storage and training time in the long run.

8 LIMITATIONS AND FUTURE WORK

The Cost of the Source Domain. In this paper, the source domain is a fingerprint database collected at the beginning. Since we focus on fingerprint updates, we assume the fingerprint database has existed. However, in practice we have to consider the cost of site survey building a fingerprint database. Fortunately, there have been many outstanding works to reduce the cost [1, 5, 9, 37]. The fingerprint database can be constructed with little human intervention by simultaneous localization and mapping (SLAM) techniques [1, 5]. Gao *et al.* integrate SLAM with inertial sensors to construct a fingerprint database semi-automatically [5]. An autonomous mapping platform is presented to collect fingerprints with a robot in [1]. Recently, deep-learning-based approaches are utilized to generate fine-grained fingerprint database leveraging unlabeled fingerprints [37, 39]. Therefore, it is possible to construct a fingerprint database as the source domain at a low cost. However, in this way fingerprint database construction and update have to be addressed independently. For future work, we will extend our work to build a deep-learning-based framework integrating fingerprint database construction and update without labeled fingerprints, making it work efficiently in a systematic way.

Wi-Fi fingerprinting with CSI (Channel State Information). Compared with RSSI, Wi-Fi CSI contains more detail of signals from physical layer which makes it a great candidate for indoor localization. Many prestigious works demonstrate that CSI works better for localization achieving decimeter accuracies [16, 30, 34, 38, 44, 45]. However, until today few commercial Wi-Fi devices make CSI available except the early release of CSItools for Intel 5300 Wi-Fi chips [8] and Atheros Ath9k series Wi-Fi chips [43]. The limitation on accessibility of CSI makes it difficult for real-world deployment [18, 50]. Therefore, in this paper we take Wi-Fi RSSIs as fingerprints and consider the fingerprint update problem raised in practice. In addition, CSI-based fingerprinting algorithms still face the same challenges as RSSI-based ones, including costly site survey and stale fingerprints over time [3, 19]. We leave further exploration of these topics for future work.

9 RELATED WORK

There are numerous studies on Wi-Fi fingerprint indoor localization [9, 25]. Due to space limitations, we focus on reviewing those works that are closely related to MTLoc.

Reducing Costly Site Survey. For fingerprint-based localization approaches, time-consuming and labor-intensive site survey has been a big hurdle in practical deployments. There are many works on reducing efforts of the site survey. Fingerprint crowdsourcing has recently been promoted to relieve the burden of the site survey by allowing unprofessional users to participate in fingerprint collection[17, 29, 36, 41, 48]. Recently, Wang *et al.* [37] propose a semi-supervised deep generative model to generate high-accuracy virtual fingerprints at a low cost. Guo *et al.* [7] propose a weak-supervised-based positioning framework to avoid high cost and low coverage due to limited labeled data. What's more, simultaneous localization and mapping (SLAM) techniques are also applied to build radio map automatically [1, 5]. MTLoc targets automatic update to reduce maintenance efforts in the long term. It is orthogonal with these works and can be put together to provide practical indoor localization services.

Inertial-Sensor-Assisted Fingerprint Update. To reduce the maintenance efforts in the long term, many works target automatic update by fusing inertial sensors [12, 35, 42]. Wu *et al.* [42] propose an automatic and continuous radio map self-updating service by accurately pinpointing mobile devices with a novel trajectory-matching algorithm. Huang *et al.* [12] propose an online radio map update scheme based on gaussian process regression (GPR) leveraging inertial sensors. Although these approaches can reduce maintenance efforts to some degree, using inertial sensors raises new issues, such as orientation estimation and additional power consumption. In contrast, MTLoc achieves achieve high accuracy in the long term without inertial sensors.

Learning-Based Fingerprint Update. Many works apply machine learning techniques by leveraging unlabeled fingerprints. Pioneering works [27, 51] propose transfer learning to update fingerprints over time. He *et al.* [10] detect the altered APs by leveraging subset sampling and updating the corresponding part of the fingerprint database with GPR. Li *et al.* [18] propose a semi-supervised update framework based on co-training, achieving room-level localization accuracy in large venues. Tian *et al.* [32] propose a single target domain adaptation model to transfer location knowledge unlabeled fingerprints. Li *et al.* [19] introduce a domain-adaptation based approach addressing signal robustness issues. Different from them, MTLoc manages to achieve high accuracy in the long term by aligning the source domain and multiple target domains with limited storage and training costs.

Unsupervised Deep Domain Adaptation. Domain adaptation targets a spectrum of applications where labeled training data from a source domain and testing data from a related but different target domain [40]. Among various approaches, unsupervised domain adaptation is the most promising one due to the reduction of reliance on target data labels. DANN [4] learns domain-invariant features by domain-adversarial training using standard backpropagation and stochastic gradient descent. CyCADA [11] adapts between domains using both pixel-level and low-level space alignment and avoids divergence before and after adaptation. DualGAN [49] employs two GANs to implement unsupervised image-to-image translation. Chen *et al.* [2] propose an unsupervised domain adaptation framework to transfer motion knowledge from one pose to another. However, the application of unsupervised deep domain adaptation in fingerprint-based localization faces linearly increasing storage and regular training costs brought by the stream of unlabeled fingerprints. MTLoc addresses the challenge by designing an automatic-updated Wi-Fi fingerprinting localization system based on unsupervised multi-target domain adaptation.

10 CONCLUSION

We carried out over-one-year experiments to study the long-term variation of APs and their Wi-Fi signals. The experimental findings reveal insights behind deteriorative accuracy of Wi-Fi fingerprinting systems in the long term, which we believe will benefit the real-world deployment of such systems. Inspired by the experimental findings, we proposed a multi-target domain adaptation model MTDAN, and designed a practical localization system MTLoc based on it. MTLoc utilizes low-cost unlabeled fingerprints to update the model with neither inertial sensor assistance nor floorplan constraints, pushing the update cost to an extreme limit. Extensive experiments demonstrate that MTLoc retains high accuracy at a low cost in various indoor environments even over two years. However, MTLoc still depends on labeled fingerprints. In future work, we will explore robust Wi-Fi fingerprinting localization without any labeled fingerprints and further improve accuracy by considering Wi-Fi CSI.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for their valuable comments and suggestions. This work was supported in part by National Natural Science Foundation of China (NSFC) under Grant 61972283, and by the key project from NSFC under Grant 61832013.

REFERENCES

[1] Roshan Ayyalasomayajula, Aditya Arun, Chenfeng Wu, Sanatan Sharma, Abhishek Rajkumar Sethi, Deepak Vasisht, and Dinesh Bharadia. 2020. Deep Learning Based Wireless Localization for Indoor Navigation. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. ACM, New York, USA, Article 17, 14 pages. https://doi.org/10.1145/3372224.3380894

[2] Changhao Chen, Yishu Miao, Chris Xiaoxuan Lu, Linhai Xie, Phil Blunsom, Andrew Markham, and Niki Trigoni. 2019. MotionTransformer: Transferring Neural Inertial Tracking between Domains. In *AAAI '19*. 8009–8016. https://doi.org/10.1609/aaai.v33i01.33018009

[3] Xi Chen, Hang Li, Chenyi Zhou, Xue Liu, Di Wu, and Gregory Dudek. 2020. FiDo: Ubiquitous Fine-Grained WiFi-Based Localization for Unlabelled Users via Domain Adaptation. In *The World Wide Web Conference* (Taipei, Taiwan) *(WWW '20)*. ACM, New York, USA, 23–33. https://doi.org/10.1145/3366423.3380091

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research* 17, 1 (Jan. 2016), 2096–2030.

[5] Chao Gao and Robert Harle. 2018. Semi-Automated Signal Surveying Using Smartphones and Floorplans. *IEEE Transactions on Mobile Computing* 17, 8 (2018), 1952–1965. https://doi.org/10.1109/TMC.2017.2776128

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS'14*. MIT Press, 2672–2680.

[7] Baoshen Guo, Weijian Zuo, Shuai Wang, Wenjun Lyu, Zhiqing Hong, Yi Ding, Tian He, and Desheng Zhang. 2022. WePos: Weak-Supervised Indoor Positioning with Unlabeled WiFi for On-Demand Delivery. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2, Article 54 (jul 2022), 25 pages. https://doi.org/10.1145/3534574

[8] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2011. Tool Release: Gathering 802.11n Traces with Channel State Information. *SIGCOMM Comput. Commun. Rev.* 41, 1 (jan 2011), 53. https://doi.org/10.1145/1925861.1925870

[9] Suining He and S-H Gary Chan. 2016. Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons. *IEEE Communications Surveys Tutorials* 18, 1 (Firstquarter 2016), 466–490. https://doi.org/10.1109/COMST.2015.2464084

[10] Suining He, Wenbin Lin, and S. H. Gary Chan. 2017. Indoor Localization and Automatic Fingerprint Update with Altered AP Signals. *IEEE Transactions on Mobile Computing* 16, 7 (2017), 1897–1910.

[11] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *International Conference on Machine Learning*. PMLR, 1989–1998.

[12] Baoqi Huang, Zhendong Xu, Bing Jia, and Guoqiang Mao. 2019. An Online Radio Map Update Scheme for WiFi Fingerprint-Based Localization. *IEEE Internet of Things Journal* 6, 4 (2019), 6909–6918. https://doi.org/10.1109/JIOT.2019.2912808

[13] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML'15*. JMLR.org, 448–456.

[14] Jin-Woo Jang and Song-Nam Hong. 2018. Indoor Localization with WiFi Fingerprinting Using Convolutional Neural Network. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, USA, 753–758.

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

[16] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. Spotfi: Decimeter level localization using WiFi. In *ACM SIGCOMM*. ACM, 269–282.

[17] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. 2014. Accurate Indoor Localization with Zero Start-up Cost. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking* (Maui, Hawaii, USA) *(MobiCom '14)*. ACM, New York, USA, 483–494. https://doi.org/10.1145/2639108.2639142

[18] Danyang Li, Jingao Xu, Zheng Yang, Yumeng Lu, Qian Zhang, and Xinglin Zhang. 2021. Train once, locate anytime for anyone: Adversarial learning based wireless localization. In *IEEE Conference on Computer Communications (INFOCOM 2021)*. IEEE, USA, 1–10.

[19] Hang Li, Xi Chen, Ju Wang, Di Wu, and Xue Liu. 2022. DAFI: WiFi-Based Device-Free Indoor Localization via Domain Adaptation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 4, Article 167 (dec 2022), 21 pages. https://doi.org/10.1145/3494954

[20] Liqun Li, Guobin Shen, Chunshui Zhao, Thomas Moscibroda, Jyh-Han Lin, and Feng Zhao. 2014. Experiencing and Handling the Diversity in Data Density and Environmental Locality in an Indoor Positioning Service. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking* (Maui, Hawaii, USA) *(MobiCom '14)*. ACM, New York, USA, 459–470. https://doi.org/10.1145/2639108.2639118

[21] Zhizhong Li and Derek Hoiem. 2018. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2018), 2935–2947. https://doi.org/10.1109/TPAMI.2017.2773081

[22] Germán Martín Mendoza-Silva, Philipp Richter, Joaquín Torres-Sospedra, Elena Simona Lohan, and Joaquín Huerta. 2018. Long-Term WiFi Fingerprinting Dataset for Research on Robust Indoor Positioning. *Data* 3, 1 (2018). https://doi.org/10.3390/data3010003

[23] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. In *ICLR*.

[24] Jiazhi Ni, Fusang Zhang, Jie Xiong, Qiang Huang, Zhaoxin Chang, Junqi Ma, BinBin Xie, Pengsen Wang, Guangyu Bian, Xin Li, and Chang Liu. 2022. Experience: Pushing Indoor Localization from Laboratory to the Wild. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking* (Sydney, NSW, Australia) *(MobiCom '22)*. Association for Computing Machinery, New York, NY, USA, 147–157. https://doi.org/10.1145/3495243.3560546

[25] Qun Niu, Tao He, Ning Liu, Suining He, Xiaonan Luo, and Fan Zhou. 2020. MAIL: Multi-Scale Attention-Guided Indoor Localization Using Geomagnetic Sequences. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2, Article 54 (Jun 2020), 23 pages. https://doi.org/10.1145/3397335

[26] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier GANs. In *International conference on machine learning*. PMLR, 2642–2651.

[27] Sinno Jialin Pan, James T. Kwok, Qiang Yang, and Jeffrey Junfeng Pan. 2007. Adaptive Localization in a Dynamic WiFi Environment through Multi-View Learning. In *Proceedings of the 22nd National Conference on Artificial Intelligence* (Vancouver, British Columbia, Canada) *(AAAI '07)*. AAAI Press, 1108–1113.

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS 2017 Workshop on Autodiff*.

[29] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. 2012. Zee: zero-effort crowdsourcing for indoor localization. In *MobiCom '12*. 293–304.

[30] Souvik Sen, Božidar Radunovic, Romit Roy Choudhury, and Tom Minka. 2012. You Are Facing the Mona Lisa: Spot Localization Using PHY Layer Information. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (Low Wood Bay, Lake District, UK) *(MobiSys '12)*. ACM, New York, USA, 183–196. https://doi.org/10.1145/2307636.2307654

[31] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NIPS*. 3483–3491.

[32] Yu Tian, Jiankun Wang, and Zenghua Zhao. 2021. Wi-Fi Fingerprint Update for Indoor Localization via Domain Adaptation. In *2021 IEEE ICPADS*. 835–842.

[33] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[34] Deepak Vasisht, Swarun Kumar, and Dina Katabi. 2016. Decimeter-level Localization with a Single WiFi Access Point. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation* (Santa Clara, CA) *(NSDI '16)*. USENIX Association, Berkeley, CA, USA, 165–178. http://dl.acm.org/citation.cfm?id=2930611.2930623

[35] Bang Wang, Qiuyun Chen, Laurence T Yang, and Han-Chieh Chao. 2016. Indoor smartphone localization via fingerprint crowdsourcing: challenges and approaches. *IEEE Wireless Communications* 23, 3 (June 2016), 82–89. https://doi.org/10.1109/MWC.2016.7498078

[36] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. 2012. No Need to War-Drive: Unsupervised Indoor Localization. In *MobiSys '12*. ACM, 197–210. https://doi.org/10.1145/2307636.2307655

[37] Jiankun Wang, Zenghua Zhao, Jiayang Cui, Yu Wang, YiYao Shi, and Bin Wu. 2021. Low-Cost Wi-Fi Fingerprinting Indoor Localization via Generative Deep Learning. In *WASA*. Springer International Publishing, 53–64.

[38] Xuyu Wang, Xiangyu Wang, and Shiwen Mao. 2017. CiFi: Deep convolutional neural networks for indoor localization with 5 GHz Wi-Fi. In *2017 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC.2017.7997235

[39] Xiangyu Wang, Xuyu Wang, Shiwen Mao, Jian Zhang, Senthilkumar CG Periaswamy, and Justin Patton. 2020. Indoor Radio Map Construction and Localization With Deep Gaussian Processes. *IEEE Internet of Things Journal* 7, 11 (2020), 11238–11249. https://doi.org/10.1109/JIOT.2020.2996564

[40] Garrett Wilson and Diane J. Cook. 2020. A Survey of Unsupervised Deep Domain Adaptation. *ACM Trans. Intell. Syst. Technol.* 11, 5, Article 51 (July 2020), 46 pages. https://doi.org/10.1145/3400066

[41] Chenshu Wu, Zheng Yang, and Yunhao Liu. 2015. Smartphones Based Crowdsourcing for Indoor Localization. *IEEE Transactions on Mobile Computing* 14, 2 (2015), 444–457. https://doi.org/10.1109/TMC.2014.2320254

[42] Chenshu Wu, Zheng Yang, Chaowei Xiao, Chaofan Yang, Yunhao Liu, and Mingyan Liu. 2015. Static power of mobile devices: Self-updating radio maps for wireless indoor localization. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, USA, 2497–2505. https://doi.org/10.1109/INFOCOM.2015.7218639

[43] Yaxiong Xie, Zhenjiang Li, and Mo Li. 2015. Precise Power Delay Profiling with Commodity WiFi. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. ACM, New York, USA, 53–64. https://doi.org/10.1145/2789168.2790124

[44] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2019. MD-Track: Leveraging Multi-Dimensionality for Passive Indoor Wi-Fi Tracking. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. ACM, New York, USA, Article 8, 16 pages. https://doi.org/10.1145/3300061.3300133

[45] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: a fine-grained indoor location system. In *Proc. of USENIX NSDI*. 71–84.

[46] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv:1505.00853 [cs.LG]

[47] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. 2021. LIMU-BERT: Unleashing the Potential of Unlabeled Data for IMU Sensing Applications. In *SenSys '21*. ACM, 220–233. https://doi.org/10.1145/3485730.3485937

[48] Zheng Yang, Chenshu Wu, Zimu Zhou, Xinglin Zhang, Xu Wang, and Yunhao Liu. 2015. Mobility Increases Localizability: A Survey on Wireless Indoor Localization Using Inertial Sensors. *ACM Comput. Surv.* 47, 3, Article 54 (April 2015), 34 pages. https://doi.org/10.1145/2676430

[49] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *2017 IEEE International Conference on Computer Vision (ICCV)* (Venice, Italy). 2868–2876. https://doi.org/10.1109/ICCV.2017.310

[50] Daqing Zhang, Dan Wu, Kai Niu, Xuanzhi Wang, Fusang Zhang, Jian Yao, Dajie Jiang, and Fei Qin. 2022. Practical Issues and Challenges in CSI-based Integrated Sensing and Communication. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*. 836–841. https://doi.org/10.1109/ICCWorkshops53468.2022.9814523

[51] Vincent Wenchen Zheng, Evan Wei Xiang, Qiang Yang, and Dou Shen. 2008. Transferring Localization Models over Time. In *Proceedings of the 23rd National Conference on Artificial Intelligence* (Chicago, Illinois, USA) *(AAAI '08)*. AAAI Press, 1421–1426.