Semi-supervised Learning with Network Embedding on Ambient RF Signals for Geofencing Services

Weipeng Zhuo*, Ka Ho Chiu*, Jierun Chen*, Jiajie Tan*
Edmund Sumpena⋆, S.-H. Gary Chan*, Sangtae Ha†, Chul-Ho Lee‡
*The Hong Kong University of Science and Technology, ⋆Johns Hopkins University
†University of Colorado Boulder, ‡Texas State University

***Abstract*—In applications such as elderly care, dementia anti-wandering and pandemic control, it is important to ensure that people are within a predefined area for their safety and well-being. We propose GEM, a practical, semi-supervised Geofencing system with network EMbedding, which is based only on ambient radio frequency (RF) signals. GEM models measured RF signal records as a weighted bipartite graph. With access points on one side and signal records on the other, it is able to precisely capture the relationships between signal records. GEM then learns node embeddings from the graph via a novel bipartite network embedding algorithm called BiSAGE, based on a Bipartite graph neural network with a novel bi-level SAmple and aggreGatE mechanism and non-uniform neighborhood sampling. Using the learned embeddings, GEM finally builds a one-class classification model via an enhanced histogram-based algorithm for in-out detection, i.e., to detect whether the user is inside the area or not. This model also keeps on improving with newly collected signal records. We demonstrate through extensive experiments in diverse environments that GEM shows state-of-the-art performance with up to 34% improvement in F-score. BiSAGE in GEM leads to a 54% improvement in F-score, as compared to the one without BiSAGE.**

I. Introduction

Many applications have been enabled by digital geofencing technology. For instance, in nursing homes for the elderly and patients [1] and medical observation for pandemic control [2], a user with an IoT device that can sense radio frequency (RF) signals stays within a designated area for a certain period of time. The user or the user's caregiver gets informed when the user gets out of the area. In constrained navigation of unmanned aerial vehicles (UAVs) [3] and logistics management [4], a UAV or a freight is restricted to move within a specific region. Otherwise, an alert will be issued. In these cases, exact locations of the user within the area are irrelevant.

A traditional approach for geofencing is to localize a user with GPS or cell tower triangulation [5]–[9]. However, it is *ineffective* in indoor environments or highly complex metropolitan environments like New York due to non-line-of-sight radio propagation or signal fading. Another approach is to exploit a network-based localization [2], [10], which relies on the IP address of a user's network device. This approach may work for relatively large space but does not work well for indoor premises requiring house-level or room-level accuracy. Furthermore, indoor localization systems [11]–[16] may be leveraged for geofencing. However, they usually require maps and collections of fingerprints, i.e., pairs of RF signals and corresponding location labels, in the localization area, which are often infeasible or labor-intensive to obtain. In other words, they would introduce privacy concerns and extra overhead in deployment, when they are used for geofencing.

To design a non-obtrusive yet highly accurate geofencing system, it is desirable to use ambient RF signals available in the region. Recent advances in machine learning can also be leveraged to that end. However, there are non-trivial technical challenges. While each RF signal record is often represented as a vector of pairs of sensed access points (APs) and received signal strength (RSS) values from them, the records need to be of *identical* length to be used for building a learning model [17]–[23]. In other words, the RF environment needs to be fairly static, which is far from reality. The APs and recorded RSS values can vary spatially and temporally. Even on the same spot, they can differ due to environmental changes. APs could also be added or removed.

With the challenges in mind, we propose **GEM** (**G**eofencing with network **EM**bedding), a semi-supervised learning system for automated IoT geofencing. As shown in Figure 1, in this system, a limited set of RF signal records, each having RSS values from ambient APs, are initially collected inside the geofencing area to build a one-class classification model for 'in-out' detection, which is to detect whether the user is inside the area (normal) or outside (abnormal). A new RF signal record is then obtained on a regular basis and fed into the one-class classification model for the in-out detection. This model is built through the following three steps in GEM.

From the initial signal records, GEM first constructs *a weighted bipartite graph*, reflecting the observation in each record while being *dynamic* to include future incoming signal records. We represent an AP as a node of one type and a record *itself* as a node of another type. Considering the inherent structure of each signal record having a list of sensed APs and their corresponding RSS values, we create edges between a node corresponding to each record and nodes corresponding to the APs detected in the record. We then assign weight values to the edges based on the RSS values. The connectivity structure in the bipartite graph not only reflects the information in each record but also captures the relevance between records. This graph representation quickly scales up to support a large number of incoming signal records over time. It is also flexible to reflect the arrivals (or departures) of new (or old) APs.

GEM next extracts *a low-dimensional vector representation* (node embedding) of each node from the weighted bipartite graph. It naturally avoids the issue with the signal records of

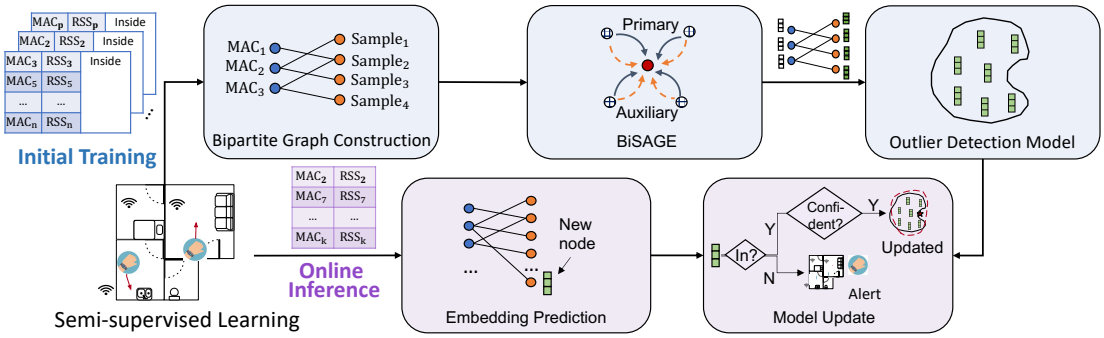


Fig. 1. A system overview of GEM.

variable size since the node embeddings are of equal length and they are used on behalf of the signal records. To this end, we propose BiSAGE, a novel **Bi**partite network embedding algorithm with **SA**mple and aggre**G**at**E**. It adopts a graph neural network architecture inspired by GraphSAGE [24], while having its own algorithmic innovations tailor-made for *weighted bipartite* graphs. Note that GraphSAGE was designed for *homogeneous* graphs where all nodes are of the same type. As shall be shown in Section V, we empirically demonstrate that GEM (with BiSAGE) outperforms its version with GraphSAGE. We also discuss how BiSAGE is different from other bipartite network embedding algorithms in Section II.

Specifically, in BiSAGE, we first propose a novel *bi-level aggregation* mechanism to differentiate embeddings for the nodes of different types and update the node embeddings based only on the ones from the nodes of the same type. To be precise, we introduce an *auxiliary* embedding for each node in addition to its *primary* node embedding. This auxiliary embedding is used as a 'carrier' for information propagation in a way that carries an aggregation of primary node embeddings from the nodes of a different type than the current node and bypasses the current one. For example, when updating the primary embeddings of the 'signal-record' nodes, the auxiliary embeddings of the 'AP' nodes are passed to their neighboring signal-record nodes, but the AP nodes' primary embeddings are left untouched, and vice versa. As shown in Figure 2, the primary embeddings of signal-record nodes 1 and 2 can communicate with each other through the auxiliary embedding of AP node 2, without affecting the primary embedding of AP node 2. Second, since the graph is a weighted graph, we employ *non-uniform* neighborhood sampling based on edge weights. Thus, a form of 'attention' is naturally introduced in the neighborhood aggregation due to the edge weights. Third, we introduce a new loss function to obtain both the primary and auxiliary embeddings per node.

Once the embeddings of initial RF signal records are obtained via the weighted bipartite graph modeling and BiSAGE, they form a training set of 'in-premises' (normal) data samples for an *enhanced histogram-based* one-class classification algorithm for in-out detection. The rationale behind the histogram-based detection algorithm is to better capture the (potentially multimodal) distribution of 'normal' data samples since the data samples, i.e., RF signal records, can exhibit quite different

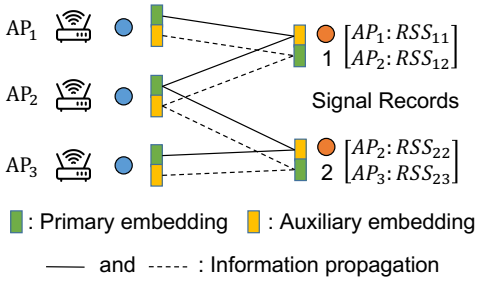


Fig. 2. In BiSAGE, the auxiliary embedding of a node serves as a carrier for information propagation of the primary embeddings of its neighbors (that are of the other node type) in the bipartite graph. The primary embeddings are the ones used for classification.

characteristics depending on where they are collected within the area. Furthermore, the training data samples are relatively small, so their representation of the geofencing area, especially its boundary, may be coarse. To cope with this, every incoming signal record (in its embedding) is used not only for in-out detection but also to augment the in-premises data for the histogram-based detection algorithm, if it is predicted to be highly confident as an 'in-boundary' data sample. This (unsupervised) data augmentation continues over time and leads to better in-out detection performance.

In summary, we have the following contributions:

- *GEM effectively learns a vector representation of each RF record via our novel embedding algorithm BiSAGE.* Since each node in the weighted bipartite graph is transformed into a low-dimensional vector space via BiSAGE, the RF records of variable length are now mapped onto the same space, so their similarities and differences can be better identified. The node embeddings generated by BiSAGE also lead to better in-out detection performance, as compared to the ones by GraphSAGE.
- *GEM is self-evolving with new RF signal records.* Upon the arrival of a new RF signal record, our histogram-based detection algorithm is used not only for in-out detection but also to augment the 'in-premises' data, if the new record is predicted to be a highly confident in-premise data sample. This self-enhancement makes the in-out detection performance keep on improving.
- *GEM significantly outperforms state-of-the-art algorithms.* We implement GEM in Android phones and evaluate its performance under various housing and RF environments.

Experiment results show that GEM achieves state-of-the-art detection performance, thanks to its adaptivity to dynamic RF environments and its self-enhancement over time. With the bipartite graph modeling and BiSAGE, GEM improves by 54% in F-score over the case without them. GEM outperforms state-of-the-art algorithms by up to 34% in F-score. Furthermore, GEM remains effective for a wide range of densities of ambient APs.

II. Related Work

Geofencing: There are few recent studies that are closely relevant to this work. SignatureHome [2] learns a geofencing area using network connectivity, such as association with a certain AP, and building a database of RF signal readings from surrounding APs. The RF signal records are converted into fixed-length vectors where missing entries are padded with arbitrarily small values to indicate nonavailability. In a similar vein, INOA [25] leverages the RF signal readings from a set of APs for area classification. It changes each variable-size record into a set of records in which each record contains RSS values for each pair of sensed APs. INOA then builds a machine learning model based on the new records. However, our system GEM does not require such a conversion of RF signal records but is able to use all the records regardless of how many RSS values are in each record, thanks to representing them in a bipartite graph and learning their (fixed-length) embeddings via BiSAGE. Furthermore, we empirically demonstrate that GEM is *superior* to SignatureHome and INOA.

Geofencing can also be considered as an indoor localization problem [11]–[16], [26]–[29]. For example, Fidora [16] collects channel state information for each location in the geofencing area and trains a classifier to predict the location of a user. SIABR [15] builds a database of fingerprints, i.e., pairs of RF signals and corresponding location labels, to train bi-directional LSTM models for location inference. However, construction of such a database is labor-intensive and time-consuming. They also require indoor floorplans to obtain exact locations of users, which may raise privacy concerns. LOCATER [29] first obtains a rough estimate of the location of the user, i.e., a predefined region covered by an AP, by recognizing the user's associated AP. It then detects which room the user is in by using the user's past behavior patterns and her potential companions (i.e., the appearance of their devices) recorded in the database. However, the user behavior analysis requires substantial logs from the past user activities, which are not available in our case. Moreover, since these indoor-localization methods merely provide an estimate of the user's location, the map of the geofencing area should be available for the location estimate to be usable for in-out detection. In contrast, our proposed system GEM only requires RF signals collected inside the area, which is *light-weight* and *privacy-preserving*. It is also efficient in detecting outliers once an out-of-boundary event happens.

Learning on heterogeneous graphs: In GEM, BiSAGE learns the embeddings of the nodes of a bipartite graph, which is a kind of heterogeneous graph. We here review recent network embedding and graph neural network algorithms for heterogeneous graphs. BiNE [30] and E-LINE [31] are network embedding algorithms for bipartite graphs, but are transductive learning algorithms to learn node embeddings from a bipartite yet *static* graph. However, BiSAGE is an inductive learning algorithm, so it can be readily used for obtaining the embeddings of new nodes (e.g., new RF signal records) that are streamed and added into the bipartite graph. In addition, GRAPE [32] is a novel algorithm for representation learning on bipartite graphs. Its main focus is to deal with missing values in the features associated with nodes for representation learning, and it merely treats the bipartite graph as a homogeneous one. However, BiSAGE uses the underlying bipartite graph *as is* to learn node embeddings from the graph without any *prior* node features.

For learning on heterogeneous graphs (beyond the bipartite graphs), the concepts of 'metapaths' have been used to guide the aggregation of information, e.g., features and embeddings, from the nodes of different types [33]–[40]. A metapath is an abstraction of a network path across the nodes of different types. For example, based on the metapath-based aggregation, HybridGNN [40] explores the importance of inter-relationship between different nodes through a randomized exploration to learn better node embeddings. However, when it comes to bipartite graphs, it is deemed *unnecessary* since there are only two types of nodes. In addition, attention mechanisms have been used to implicitly specify weights to edges into the nodes of different types [41]–[43]. For example, HIVEN [43] is developed for a heterogeneous graph where each node may have multiple types of relationship with others. It learns type-wise embeddings of each node, which are then combined to obtain the final node embedding per node using an attention mechanism where the attention weights are learned through training. In contrast, by definition, our underlying graph is a *weighted* graph, which does not necessarily require learning the attention weights. Furthermore, BiSAGE enables GEM to achieve (almost) ideal performance without such an additional mechanism.

Outlier detection: The in-out detection from the geofencing area can be cast as an outlier detection [20], [44]–[46] problem where any RF signal records obtained *outside* the area are to be detected as outliers. DBSCOUT [45], which is built on DBSCAN, treats a point as an outlier if it is detected outside the hypersphere (dense region) with a predefined radius. It is, however, tested only on two- and three-dimensional data. RDAE [46] leverages two layers of autoencoders and decomposes input data into normal and outlier data samples for training. In contrast, GEM, especially our enhanced histogram-based detection algorithm, does not require any *labeled* outliers as input data samples, but is able to detect outliers using the training dataset that consists of only normal data samples. Furthermore, its performance is extensively evaluated in comparison with several popular outlier detection algorithms, such as feature bagging, isolation forest, and local outlier factor.

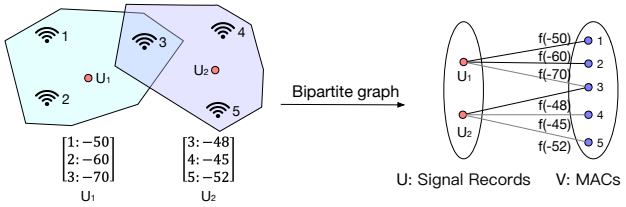


Fig. 3. An illustrating example for two sensing events with five ambient APs.

III. GEM: Initial Training

In this section, we explain the details of the three integral components of GEM to build a one-class classification model.

A. Weighted Bipartite Graph Model

It has been a common practice to represent the RF signal data in a matrix form (or vectors of equal length), when they are leveraged for various applications such as indoor localization [47] and floor identification [48]. Each signal record (sample) is a vector of RSS values from surrounding APs that are indicated by their medium access control (MAC) addresses and becomes a column of the matrix.[1] In other words, each RSS value associated with a different MAC address is for a row of the matrix, so each row index corresponds to a different MAC address. In the matrix representation, however, there is a 'missing-value' problem in that some entries in the matrix lack RSS values, as the samples are of variable length (i.e., the number of detected MAC addresses can be different per sample). While the missing entries are generally filled with arbitrarily small values, this ad-hoc data imputation potentially leads to incorrect representation.

On the other hand, as recently used in [31] for a floor classification application, the variable-length RF signal records can be modeled as a weighted bipartite graph, where the measured signal information is preserved *without* any ad-hoc data imputation. Specifically, for each RF signal record, this record itself becomes a node of one type and the sensed MAC addresses in the record become nodes of the other type, with undirected edges connecting the 'record' node and the 'MAC' nodes. Each edge is assigned a weight that is determined as a function of the RSS value from its corresponding MAC address in the record.

Observe that each RF signal record has a set of pairs of sensed MAC addresses and their corresponding RSS values. For each record (sample) u, we denote by RSS_{uv} the RSS value from a MAC address, say MAC_v, which appears in the record. Let U be a set of nodes for the signal records and V be a set of nodes for the sensed MAC addresses. We then define a weighted bipartite graph $\mathcal{G}=(U,V,\mathcal{E},\boldsymbol{w})$, where $\mathcal{E}$ is a set of edges with edge $e_{uv}\in\mathcal{E}$ denoting the edge between $u\in U$ and $v\in V$, and $\boldsymbol{w}$ is a set of edge weights with weight $w_{uv}\in\boldsymbol{w}$ of edge e_{uv}, which is defined as

$$w_{uv} := f(\text{RSS}_{uv}), \tag{1}$$

[1]Note that each AP can have one or more MAC addresses associated with its transceivers, and the MAC addresses are the ones actually recorded in each RF signal record. We use MAC addresses instead of APs for clarity.

where f is a function of RSS_{uv} such that $f(\text{RSS}_{uv})>0$ for all RSS_{uv}. Note that edge e_{uv} indicates the presence of MAC_v in record u. In other words, two nodes u and v are connected if there exists a measured value of RSS_{uv} between them, with edge weight w_{uv}. In the example illustrated in Figure 3, RF signal record u_1 is only connected to MACs 1–3, while u_2 is connected to MACs 3–5.

In GEM, we also use the following weight function:

$$f(\text{RSS}_{uv}) := \text{RSS}_{uv} + c, \tag{2}$$

where c is a constant such that $c>\max\{|\text{RSS}_{uv}|,\forall u,v\}$ [31]. It is also worth noting that this bipartite graph representation can be easily extended to include the arrivals of new RF signal records and the changes in ambient APs (or their MAC addresses).

B. BiSAGE

From the weighted bipartite graph $\mathcal{G}$, we next learn node embeddings of equal length to be used to build a one-class classification model for in-out detection. To this end, we propose BiSAGE, which is a non-trivial extension to GraphSAGE [24] with a novel bi-level aggregation and a non-uniform neighborhood sampling.

Given a target node whose embedding is to be learned, there are two steps in the aggregation of embeddings from its (possibly multi-hop) neighbors. We sample nodes from the neighborhood of the target node according to a given probability distribution. Then, we aggregate the information (embeddings) from sampled neighboring nodes towards the target node (to update its embedding).

In the bipartite graph, edges are assigned weights that are a function of sensed RSS values. Thus, to determine which neighbors to sample, intuitively, the higher the sensed RSS value between the node and its neighbor, the more likely the neighbor should be chosen for the information aggregation. Thus, we design a non-uniform neighbor sampling based on edge weights. Without loss of generality, suppose that $u\in U$ is the target node to learn its embedding. Let $N(u)$ be the set of neighbors of u in the bipartite graph. The probability that $v\in N(u)$ is chosen for the aggregator function is defined as

$$\Pr(v) = \frac{w_{uv}}{\sum_{v'\in N(u)} w_{uv'}}.$$

After sampling the neighbors of the target node, we need to aggregate the information from the sampled neighbors to the target node. If the underlying graph is a homogeneous graph, the information aggregation can be done for each node in the same way [24]. However, in the bipartite graph (a heterogeneous graph), nodes of different types should be processed differently. Thus, we propose a novel bi-level aggregation mechanism for bipartite graphs.

In the bi-level aggregation, we introduce a new 'auxiliary' embedding for each node. We consider the original embedding of each node as its 'primary' embedding. In other words, each node has two embeddings, namely primary embedding and auxiliary embedding. The auxiliary embedding at each

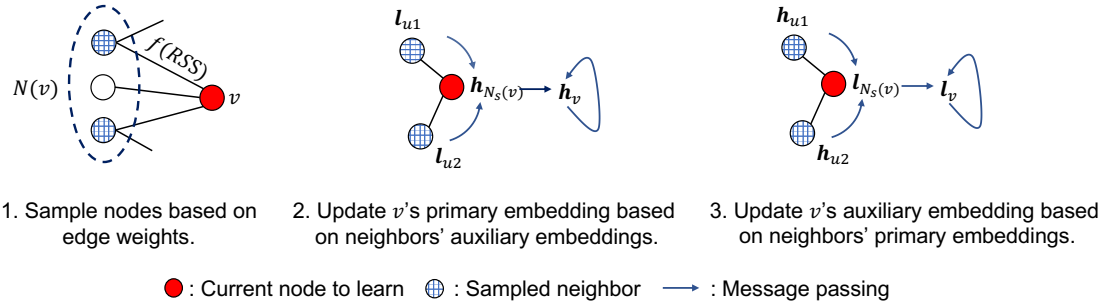


Fig. 4. To learn a node's embedding, BiSAGE first samples its neighbors and then aggregates the information from them via their auxiliary embeddings.

node now serves as a 'carrier' for the information propagation when the target node is of different type. For example, if the target node is a 'signal-record' node, its neighbors – 'sensed MAC' nodes – use their auxiliary embeddings to propagate the information.

As shown in Figure 4, for the target node, we sample its neighbors based on their edge weights for the information aggregation. The target node's primary embedding is then updated based on its sampled neighbors' auxiliary embeddings that carry the information (embeddings) from the nodes of the same type. On the other hand, the target node's auxiliary embedding is updated based on its sampled neighbors' primary embeddings to carry the information. Note that the *auxiliary* embedding at each node constrains the influence of the information within the nodes of the same type, while the information can still flow over the graph.

For node $i \in U \cup V$, let $\boldsymbol{h}_i$ and $\boldsymbol{l}_i$ be its d-dimensional primary and auxiliary embeddings, respectively. We denote by $N_s(i)$ the *sampled* neighborhood of i. Let k be the current aggregation step and K be the total number of steps. In the k-th round of the aggregation, we need to update both primary and auxiliary embeddings for each node. To update the primary embedding of i, we need to aggregate the information through its neighbors' auxiliary embeddings. Thus, we have

$$\boldsymbol{h}^k_{N_s(i)} \longleftarrow \text{AGGREGATE}(\boldsymbol{l}^{k-1}_j, \forall j \in N_s(i)), \tag{3}$$

$$\boldsymbol{h}^k_i \longleftarrow \sigma(\boldsymbol{W}^k_h \cdot \text{CONCAT}(\boldsymbol{h}^{k-1}_i, \boldsymbol{h}^k_{N_s(i)})), \tag{4}$$

where AGGREGATE$(\cdot)$ is an aggregator function, e.g., MEAN$(\cdot)$ or MAX$(\cdot)$, $\boldsymbol{h}_{N_s(i)}$ is a d-dimensional temporary vector to store the aggregation result from the sampled neighborhood of i, CONCAT$(\cdot)$ is a concatenation function, $\boldsymbol{W}^k_h$ is a learnable weight matrix for primary embedding, and σ is a nonlinear function. Here we introduce the superscript k to indicate the k-th round of the aggregation. Similarly, to update the auxiliary embedding of i, we need to aggregate the information from its neighbors' primary embeddings. Thus, we have

$$\boldsymbol{l}^k_{N_s(i)} \longleftarrow \text{AGGREGATE}(\boldsymbol{h}^{k-1}_j, \forall j \in N_s(i)), \tag{5}$$

$$\boldsymbol{l}^k_i \longleftarrow \sigma(\boldsymbol{W}^k_l \cdot \text{CONCAT}(\boldsymbol{l}^{k-1}_i, \boldsymbol{l}^k_{N_s(i)})), \tag{6}$$

where $\boldsymbol{l}_{N_s(i)}$ is another temporary vector to store the aggregation result from the sampled neighborhood of i, and $\boldsymbol{W}^k_l$ is another learnable weight matrix for the auxiliary embedding.

After completing the k-th round of the aggregation, h^k_i and $\boldsymbol{l}^k_i$ are then normalized by their ℓ_2 norms, respectively, as follows:

$$\boldsymbol{h}^k_i = \frac{\boldsymbol{h}^k_i}{||\boldsymbol{h}^k_i||_2}, \text{ and } \boldsymbol{l}^k_i = \frac{\boldsymbol{l}^k_i}{||\boldsymbol{l}^k_i||_2}, \tag{7}$$

which are the embeddings to be used for $(k+1)$-th iteration. After K iterations, the final primary and auxiliary embeddings of node i are $\boldsymbol{h}^K_i$ and $\boldsymbol{l}^K_i$, respectively. Initially, $\boldsymbol{h}^0_i$ and $\boldsymbol{l}^0_i$ are chosen randomly.

For our choice of the aggregator function, we observe that the higher the edge weight (based on the RSS value), the closer the two nodes having the edge should be. In other words, the information propagating through edges with higher weights should be considered more important during the aggregation. Thus, we use the following weighted aggregator function for $\boldsymbol{h}^k_{N_s(i)}$ in Equation (3):

$$\boldsymbol{h}^k_{N_s(i)} := \sum_{j \in N_s(i)} \left(\frac{w_{ji}}{\sum_{j' \in N_s(i)} w_{j'i}} \boldsymbol{l}^{k-1}_j \right). \tag{8}$$

Similarly for $\boldsymbol{l}^k_{N_s(i)}$ in Equation (5).

Next, we explain the details of the training process. A standard approach to training a model for network embedding and representation learning on a graph is to leverage random walks. They are used to learn embeddings such that the nodes that appear in the sequence of nodes visited by the same walk should be close to each other in the embedding space [24], [49]. Let $\{X_k\}^n_{k=0}$ be a finite sequence of nodes visited by a random walk when it starts from node $x_0 \in U \cup V$, i.e., $X_0 = x_0$. Given the current node of the random walk, say $X_k = x_k$, the next node x_{k+1} is chosen from the neighbors of x_k according to a certain distribution. As in GraphSAGE [24], a popular example is to use a uniform distribution for choosing the next node, i.e., the next node is chosen from the neighbors of the current node *uniformly at random*. However, our graph is a *weighted* (bipartite) graph, where each edge weight is proportional to its corresponding measured RSS value, implying that a higher RSS value generally indicates a closer distance between the IoT device and the AP. Thus, in BiSAGE, we use a weighted random walk, whose transition probabilities are given by

$$\Pr\left(X_{k+1} = x_{k+1} | X_k = x_k\right) = \frac{w_{x_k x_{k+1}}}{\sum_{x' \in N(x_k)} w_{x_k x'}}.$$

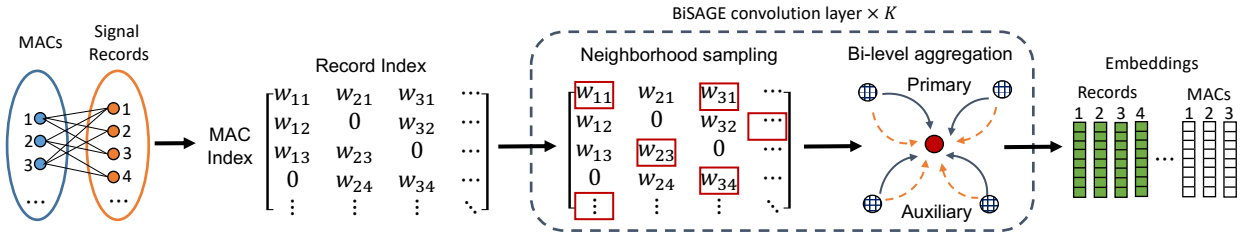


Fig. 5. An overall procedure from constructing a weighted bipartite graph to obtaining node embeddings via BiSAGE.

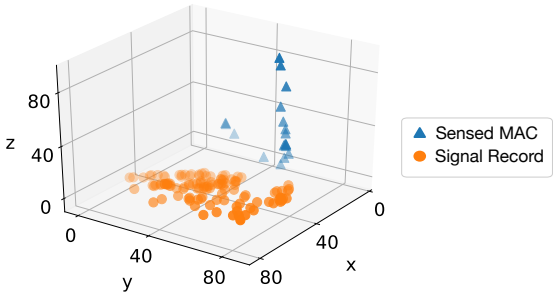


Fig. 6. A visualization of our learned embeddings.

In addition, we need to specify a loss function to minimize in learning the two sets of weight matrices $\{\boldsymbol{W}_h^k\}_{k=1}^K$ and $\{\boldsymbol{W}_l^k\}_{k=1}^K$ for primary and auxiliary embeddings, respectively. The rationale behind our loss function is that the primary embedding of one node should be close to the auxiliary embedding of its neighbor visited by the random walk, as they encode the information of nodes of the same type, and vice versa. Let x and y be two neighboring nodes that appear as the nodes consecutively visited by the random walk. Let $\boldsymbol{h}_x$, $\boldsymbol{l}_x$, $\boldsymbol{h}_y$, and $\boldsymbol{l}_y$ be their corresponding primary and auxiliary embeddings, respectively. Specifically, we use the following loss function to minimize in learning the weight matrices $\{\boldsymbol{W}_h^k\}$ and $\{\boldsymbol{W}_l^k\}$ to be used to obtain the primary and auxiliary embeddings of each node:

$$J_{\mathcal{G}} := -\log\left[\sigma(\boldsymbol{h}_x \cdot \boldsymbol{l}_y)\sigma(\boldsymbol{l}_x \cdot \boldsymbol{h}_y)\right] - K_N \mathbb{E}_{z\sim \Pr(z)}\left[\log\left[\sigma(-\boldsymbol{h}_x \cdot \boldsymbol{l}_z)\sigma(-\boldsymbol{l}_x \cdot \boldsymbol{h}_z)\right]\right], \quad (9)$$

where $\boldsymbol{a} \cdot \boldsymbol{b}$ is the inner product between $\boldsymbol{a}$ and $\boldsymbol{b}$, $\sigma(x) := 1/(1+\exp(-x))$, K_N is the number of 'negative' samples, and the expectation $\mathbb{E}$ is with respect to z drawn according to $\Pr(z)$, $z \in U \cup V$.

Our loss function in Equation (9) is based on the negative sampling technique [24], [50], but defined in a way that is suitable for our bi-level aggregation on a weighted bipartite graph. The first term of the loss function encourages nodes of the same type that co-occur in the same random walk to stay close to each other in the embedding space. Recall that the auxiliary embedding of node y encodes the information of its neighbors, which have the same type as node x. On the other hand, the second term uses K_N (negative) sampled nodes drawn from the graph and forces them to separate apart from node x, since the sampled nodes are more likely far from the current node x. We use $K_N = 4$ and $\Pr(z) \propto \deg_z^{3/4}$, where $\deg_z$ is the degree of node z [24], [50] .

In Figure 5, we summarize the overall process from constructing a weighted bipartite graph to obtaining node embeddings of the graph via BiSAGE. To show the effectiveness of our learned node embeddings, we use a visualization tool called t-SNE [51] to visualize the embeddings. We collect RF signal records inside a room and process them with BiSAGE. As shown in Figure 6, nodes of the same type generally stay together while being separated from nodes of different types. In addition, as shall be validated in Section V, the learned embeddings preserve the relevance among the nodes of the same type, i.e., their distance in the embedding space preserves their physical distance, which allows us to detect outliers based on their proximity information.

The bi-level aggregation mechanism is summarized in Algorithm 1. Its time complexity can also be analyzed as follows. Let N_s be the number of sampled neighbors for each node, i.e., $N_s = |N_s(i)|$ for all i. Note that it is a hyperparameter, whose value is normally chosen between 10 and 25. Lines 4 and 5 take $O(dN_s)$ operations each, where d is the embedding dimension. Here we use the weighted average in Equation (8) as an aggregate function. Each of Lines 6 and 7 involves a matrix multiplication with $\boldsymbol{W}^k \in \mathbb{R}^{d\times 2d}$, which has the complexity of $O(d^2)$. Lines 8 and 9 perform normalization of a d-dimensional embedding each, which has the complexity of $O(d)$. Therefore, the time complexity of Algorithm 1 is $O\left(K(|U|+|V|)(dN_s+d^2)\right)$. It is worth noting that the matrix multiplication is often done in parallel on GPUs, and we also use the minibatch training strategy as proposed in GraphSAGE [24]. Thus, the complexity would be much lower in practice.

C. In-Out Detection

We next explain our enhanced histogram-based outlier detection model that is built on the primary embeddings of RF signal records for in-out detection, to detect whether the user is inside the area (normal) or outside (outlier). Specifically, we adopt and *enhance* the histogram-based algorithm [17] due to its simplicity (fast training) and effectiveness in capturing a hidden distribution in the feature space.[2] It leverages the idea that the feature vectors from normal data exhibit similar patterns in the feature space, as long as they well represent the data. For each feature (each dimension of the feature vector),

[2]Recall that RF signal records can be quite different depending on where they are collected within the area, thereby making the distribution potentially multimodal. A histogram-based approach is suited for capturing the multi-modality in the distribution.

ALGORITHM 1: BiSAGE: Bi-level Aggregation.

Input: Bipartite graph $\mathcal{G}$; Total number of aggregation layers K.
Output: Primary and auxiliary embeddings $\boldsymbol{h}_i$ and $\boldsymbol{l}_i$ for all $i \in U \cup V$.

Initialize $\boldsymbol{h}_i$ and $\boldsymbol{l}_i$ for all $i \in U \cup V$.
for $k = 1, 2, \ldots, K$ **do**
  for $i \in U \cup V$ **do**
    /* Neighborhood aggregation */
    $\boldsymbol{h}^k_{N_s(i)} := \text{AGGREGATE}(\boldsymbol{l}^{k-1}_j, \forall j \in N_s(i))$
    $\boldsymbol{l}^k_{N_s(i)} := \text{AGGREGATE}(\boldsymbol{h}^{k-1}_j, \forall j \in N_s(i))$
    /* Embedding update */
    $\boldsymbol{h}^k_i := \sigma(\boldsymbol{W}^k_h \cdot \text{CONCAT}(\boldsymbol{h}^{k-1}_i, \boldsymbol{h}^k_{N_s(i)}))$
    $\boldsymbol{l}^k_i := \sigma(\boldsymbol{W}^k_l \cdot \text{CONCAT}(\boldsymbol{l}^{k-1}_i, \boldsymbol{l}^k_{N_s(i)}))$
    /* Normalization */
    $\boldsymbol{h}^k_i := \frac{\boldsymbol{h}^k_i}{||\boldsymbol{h}^k_i||_2}$
    $\boldsymbol{l}^k_i := \frac{\boldsymbol{l}^k_i}{||\boldsymbol{l}^k_i||_2}$
  end
end
return $\boldsymbol{h}_i, \boldsymbol{l}_i,\ i \in U \cup V$.

it builds a histogram of frequencies obtained from the feature vectors of training (normal) data.

We below explain the details of the histogram-based algorithm when used with the primary embeddings of signal records (initial normal data), and then present our enhancement. Suppose that there are n d-dimensional primary embeddings $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n$. Let $h_{i,j}$ be the j-th element (or dimension) of the i-th embedding. We first obtain the maximum value, i.e., $\Delta^u_j := \max_i\{h_{i,j}\}$, and the minimum value, i.e., $\Delta^l_j := \min_i\{h_{i,j}\}$, from the j-th element values of the n embeddings. We then create m bins, with each bin of width $(\Delta^u_j - \Delta^l_j)/m$. Since the j-th element value of each embedding $\boldsymbol{h}_i$ falls into one of the bins, we can construct a histogram based on the frequency count of each bin. Note that there is a separate histogram for each dimension j.

Given the d histograms, when a new (primary) embedding is available, i.e., a new RF signal record is available and its corresponding primary embedding is obtained (via BiSAGE), we can compute its 'outlier' score and determine whether it is an outlier by comparing the score with a threshold value. The score function and the threshold value are given as follows. For any given embedding, say $\boldsymbol{h} := [h_1, h_2, \ldots, h_d]$, its 'outlier' score is calculated by

$$H(\boldsymbol{h}) = \sum_{j=1}^{d} \log\left(\frac{1}{\text{hist}_j(h_j)}\right), \tag{10}$$

where $\text{hist}_j(h_j)$ is the frequency count of the bin where h_j belongs for each dimension j. The *higher* the score, the more likely it is an outlier. As for the threshold value, we compute the outlier scores of the n primary embeddings of the initial normal data using Equation (10). We then use the min-max normalization to normalize the scores into the range $[0, 1]$, and sort them in decreasing order. To be precise, let $\bar{H}(\boldsymbol{h}_1), \bar{H}(\boldsymbol{h}_2), \ldots, \bar{H}(\boldsymbol{h}_n)$ denote the normalized scores, and let $\bar{H}(\boldsymbol{h}_{[1]}), \bar{H}(\boldsymbol{h}_{[2]}), \ldots, \bar{H}(\boldsymbol{h}_{[n]})$ denote the *sorted*, normalized scores, where the subscript $[i]$ indicates the index of an embedding that leads to the i-th highest score. With γ defined as a 'contamination' factor and $i^* := n \times \gamma$, the threshold value, denoted by τ, for outlier detection is originally set to $\tau := \bar{H}(\boldsymbol{h}_{[i^*]})$ [17].

While the contamination factor γ can be used to control the level of the sensitivity in detecting outliers, the threshold value τ is highly dependent on the (initial) data size n and the choice of γ. In our geofencing system, we aim to augment the normal data with newly measured RF records. Since the data size keeps on increasing, however, it can lead to changes in the threshold value τ and negatively affect the performance of outlier detection. For example, if the scores of recent records are somehow relatively much higher than the old ones, then they may boost up the threshold value τ, which would lead to misclassifications of outlier (outside) records as normal (in-premises) ones, and vice versa. Thus, there is a need for an enhanced algorithm to make this outlier detection adaptive in incorporating new records.

We first reduce the dependence of choosing the threshold value τ on the data size. We also observe that if the outlier scores for normal and abnormal records can be smoothed and separated further apart, it would be easier to set the threshold value. Thus, we adopt to use the softmax function with a scaling factor T. For any given embedding $\boldsymbol{h}$, its outlier score is now obtained by

$$S_T(\boldsymbol{h}) := \frac{\exp(\bar{H}(\boldsymbol{h})/T)}{\exp(\bar{H}(\boldsymbol{h})/T) + \exp((1 - \bar{H}(\boldsymbol{h}))/T)}. \tag{11}$$

Note that S_T is in the form of a Boltzmann distribution (or Gibbs distribution) in statistical physics, where T is the thermodynamic temperature.

The intuition behind the transformation with the softmax function is that the outlier scores of *normal* samples (i.e., the embeddings of RF signal records obtained *inside* the geofencing area) get quite close to each other. Similarly for *abnormal* samples, which are the embeddings of the ones measured *outside* the area, including those collected just outside the boundary (see Section VI for further discussion on such abnormal samples). In addition, the scaling factor T further separates the former apart from the latter. Therefore, we use the new score function S_T in Equation (11) to compute the outlier score of a new sample $\boldsymbol{h}$, i.e., the (primary) embedding of a new RF signal record, and determine it is an outlier if

$$S_T(\boldsymbol{h}) > \tau_u, \tag{12}$$

where τ_u is our new threshold value. The scaling parameter T and the new threshold value τ_u are considered as hyper-parameters to be optimized in the learning process. As shall be shown in Section V, we empirically demonstrate that the rescaling method with the softmax function in Equation (11) for the histogram-based outlier detection indeed improves the

in-out detection performance.

IV. GEM: Online Inference and Update

Once our outlier detection model is built on the primary embeddings of normal RF signal records (initial training data), GEM performs in-out detection (or outlier detection) whenever a new RF signal record is available. The IoT device of the user periodically records RSS values from ambient APs, and they are fed into GEM for the in-out detection. In what follows, we first explain how to obtain the primary embeddings of the new RF signal records in an online manner, and then present how to use the new records to perform in-out prediction and further update our outlier detection model over time.

A. Embedding Prediction

The primary and auxiliary embeddings of every node in the weighted bipartite graph are learned in the training process. When a new RF signal record becomes available, it is added into the graph as a new 'signal-record' node, say r, and its edges are created with the 'MAC' nodes that appear in the record. Some MAC nodes may also be newly added, if they are new ones to the graph. Then, edge weights are determined based on recorded RSS values according to Equations (1) and (2).[3] Once node r is added into the graph, its primary embedding $\boldsymbol{h}_r$ and auxiliary embedding $\boldsymbol{l}_r$ are obtained according to Equations (3)–(7), along with the learned weight matrices $\{\boldsymbol{W}_h^k\}$ and $\{\boldsymbol{W}_l^k\}$. In other words, node r aggregates the information (embeddings) from its sampled neighbors, and this aggregation process is repeated K times. That is, node r is able to attain the knowledge from all nodes within the K-hop neighborhood, and thus its embeddings are determined by their embeddings.

B. In-out Prediction

Given the learned primary embedding $\boldsymbol{h}_r$ of the new signal record r, we need to decide whether it is an outlier or not. To this end, we first calculate its raw outlier score $H(\boldsymbol{h}_r)$ as in Equation (10) and normalize it through the min-max normalization to obtain the normalized score $\bar{H}(\boldsymbol{h}_r)$ (see Section III-C for more details). We then obtain its final outlier score $S_T(\boldsymbol{h}_r)$ as in Equation (11) and decide whether it is an outlier or not according to Equation (12).

C. Online Model Update

As illustrated in Figure 1, if a new signal record is predicted to be an outlier by our outlier detection model, it triggers an alert. On the other hand, if it is predicted to be a normal one – the record measured inside the geofencing area, we further check how confident the prediction is. Then, if it can be considered as a highly confident normal (in-premises) sample, we update our outlier detection model.

To this end, we introduce another yet more strict threshold value τ_l ($< \tau_u$) to filter out the 'normal' predictions with *low* confidence. Specifically, when the score of a new sample $\boldsymbol{h}_r$ – the primary embedding of a new record – is less than the threshold value τ_u (see Equation (12)), we say that it is a *highly confident* normal sample if it also satisfies $S_T(\boldsymbol{h}_r) < \tau_l$. Then, we use the new embedding $\boldsymbol{h}_r$ to recalculate the d histograms used in Equation (10), thereby updating our outlier detection model. The threshold value τ_l is again considered as a hyperparameter. As shall be shown in the next section, we empirically confirm that this online model update scheme indeed improves the in-out detection performance.

It is worth noting that the concept of leveraging the 'predicted' labels of originally unlabeled samples to enhance a model has been around in the literature [52], [53]. Such predicted labels are often called *pseudo labels*. It was first shown in [52] that for the data samples naturally belonging to different clusters, the pseudo labels of the (originally unlabeled) samples can improve the model performance substantially.

D. Summary and Complexity Analysis

We summarize the entire inference process in Algorithm 2 and provide its complexity analysis as follows. Observe that Line 1 is a $O(1)$ operation since a new signal record contains a limited number of MACs (normally less than 100 MACs). Also, Line 2 has the complexity of $O\left(K(dN_s+d^2)\right)$ as can be seen from the complexity analysis of Algorithm 1, where d is the embedding dimension. Line 3 requires $O(d)$ operations as the outlier score $S_T(\boldsymbol{h}_r)$ is computed along each embedding dimension. In addition, the complexity of Lines 4–8 is dominated by the model-update operation in Line 7 which has the complexity of $O(d|U|)$. It is because d histograms are recalculated based on all in-boundary signal records, including the new highly confident signal record, and the record size is bounded by $|U|$. Note that only a small portion of new records would be used to update the model in practice. Therefore, the time complexity of Algorithm 2 is $O\left(K(dN_s+d^2)+d|U|\right)$.

ALGORITHM 2: GEM: Online Inference.

Input: New signal record $\boldsymbol{r}$; Bipartite graph $\mathcal{G}$.
Output: *IN* or *OUT* for $\boldsymbol{r}$.

1 Connect $\boldsymbol{r}$ into $\mathcal{G}$.
2 Obtain $\boldsymbol{r}$'s primary embedding $\boldsymbol{h}_r$ according to Equations (3)–(7).
3 Calculate outlier score $S_T(\boldsymbol{h}_r)$ as in Equation (11).
4 **if** $S_T(\boldsymbol{h}_r) > \tau_u$ **then**
5 | **return** *OUT*
else
6 | **if** $S_T(\boldsymbol{h}_r) < \tau_l$ **then**
7 | | Update d histograms.
| **end**
8 | **return** *IN*
end

V. Performance Evaluation

In this section, we present extensive experiment results to demonstrate the efficacy of GEM. We evaluate its system-level

[3] It is possible that the new signal record only contains the MAC addresses that have *never* been seen before. In this case, it may be the one collected far from the geofencing area, and thus we treat it as an outlier, raising an alert.

performance by comparing it with state-of-the-art algorithms. We also empirically demonstrate the effectiveness of each system component.

Experiment setup: We developed an Android application to collect RSS values from ambient APs and notify if the user is out of the area. The application continuously collects RF signal records and uploads them to a server that performs in-out detection based on the gathered data.

We recruited 10 volunteers in the experiments. A user carries a smartphone such as Samsung S7, Huawei Nova 6 or Xiaomi Mix 3, or a smartwatch, i.e., alps DM20, in a typical home setting, e.g., a single-room dorm, a small or large apartment, or a two-story house. The housing area ranges from 10 m^2 to 200 m^2. Each user is asked to walk around the *inner* perimeter of the house for just a few minutes (i.e., 5-10 minutes) for initial training. Then, the user can behave as he/she wishes, i.e., staying inside or moving outside of the house for testing. The whole process lasts for about three hours for each user. To further evaluate the performance of GEM under different environmental factors, we also test GEM with the signal records collected for three days around our lab.

We use the following baseline parameters for GEM, which were obtained through hyperparameter tuning. The learning rate is 0.003, the embedding dimension is 32, the offset c is 120 dBm, the scaling factor T is 0.06, the in-out threshold τ_u is 0.005, and the updating threshold τ_l is 0.001. All experiments were conducted on a server that operates on Ubuntu 18.04 and has an Intel Core i9-9900X with 10 cores at 3.5 GHz, 64-GB memory, and an Nvidia 2080Ti GPU.

Geofencing algorithms for performance comparison: We consider the following state-of-the-art algorithms for performance evaluation. By noting that GEM consists of BiSAGE and our outlier detection algorithm (dubbed 'OD'), we also consider other network embedding and outlier detection algorithms, each of which is used together with OD or BiSAGE.

- SignatureHome [2]: It builds a 'home signature' database for the geofencing area, containing the union of the detected MACs from all RF signal records collected inside the area and the IP address of the user's associated AP. When it comes to online inference, for each newly collected signal record, it checks whether the connected IP exists in the home signature (assigned a weight) and calculates the overlap ratio of MACs between the new record and the home signature (assigned another weight). If the total weight is higher than a threshold, the record is inside the area.
- INOA [25]: It trains an ensemble of base learners where each base learner learns a hypersphere based on RF signal records from each pair of APs. For inference, a new RF signal record is decomposed into corresponding pairs of APs, which are then fed into the base learners to obtain an outlier score. If it is higher than a threshold, the record is outside the area.
- GraphSAGE [24] + OD: GraphSAGE replaces BiSAGE in GEM to obtain node embeddings from our weighted bipartite graph by treating the graph as a *homogeneous* graph. We then use the learned embeddings to build an outlier detection model as illustrated in Section III-C. For inference, upon the arrival of a new RF signal record, we first obtain its embedding via GraphSAGE and then perform outlier detection and model update as explained in Section IV-B and Section IV-C, respectively.
- Autoencoder [54] + OD: Autoencoder uses an encoding and decoding process to learn (low-dimensional) embeddings from a matrix of signal records, where the missing entries are filled with small values. The embeddings generated by the autoencoder replace those generated by BiSAGE and the rest follows as in GraphSAGE + OD.
- Multidimensional scaling (MDS) [55] + OD: MDS learns the embeddings from a matrix of pairwise distances calculated from signal records, where missing entries are padded with small values. The embeddings generated by MDS replace those generated by BiSAGE and the rest follows as in GraphSAGE + OD.
- BiSAGE + feature bagging [56]: Feature bagging designs a framework to fuse the results from different outlier detection algorithms based on different subsets of features, which are the embeddings generated by BiSAGE. When a new RF signal record is available for inference, its embedding is first learned by BiSAGE and then goes through the feature bagging framework to decide whether it is an outlier.
- BiSAGE + isolation forest (iForest) [57]: iForest finds outliers with the rationale that outliers are easier to 'isolate' from normal (in-premises) samples. It also uses the embeddings generated by BiSAGE. The rest follows as in BiSAGE + feature bagging.
- BiSAGE + local outlier factor (LOF) [58]: LOF finds outliers by observing that the distance from an outlier to a neighbor group is larger than the distance inside the neighbor group. It also uses the embeddings generated by BiSAGE. The rest follows as in BiSAGE + feature bagging.

The algorithms with BiSAGE use the embeddings generated by BiSAGE as input and perform in-out detection. The algorithms with OD first generate the embeddings on their own and carry out in-out detection with our outlier detection algorithm. For SignatureHome and INOA, we set their parameters as described in [2], [25]. For autoencoder, its best results, which are obtained using four layers of 1-D convolution with the ReLU activation function, are used for performance comparison. For MDS, we follow the convention with $1-$cosine similarity as pairwise distance.

Performance metrics: We use precision P, recall R, and F-score to evaluate classification results in our experiments. Let TP, FP, and FN be the number of true positives, the number of false positives, and the number of false negatives, respectively. Then, we have $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, and $F = \frac{2PR}{P+R}$. To better understand the system performance, we use P_{in}, R_{in} and F_{in} for in-premises detection in which case in-premises samples are treated as positive samples, and P_{out}, R_{out} and F_{out} for outlier detection in which case outside samples are considered positive.

TABLE I

PERFORMANCE COMPARISON WITH STATE-OF-THE-ART ALGORITHMS. THE ENTRY IS IN THE FORM OF MEAN (MIN, MAX). OD INDICATES OUR HISTOGRAM-BASED OUTLIER DETECTION ALGORITHM, AND BISAGE INDICATES OUR NETWORK EMBEDDING ALGORITHM.

Algorithms	In-premises detection			Outside detection		
	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	F_{out}
GEM (BiSAGE + OD)	**0.98** (0.94, 1.00)	**0.99** (0.94, 1.00)	**0.98** (0.97, 1.00)	**0.98** (0.88, 1.00)	**0.97** (0.88, 1.00)	**0.97** (0.94, 1.00)
SignatureHome	0.98 (0.86, 1.00)	0.97 (0.92, 1.00)	0.97 (0.88, 0.98)	0.80 (0.62, 0.90)	0.86 (0.76, 0.95)	0.83 (0.70, 0.93)
INOA	0.95 (0.75, 0.98)	0.82 (0.57, 0.88)	0.88 (0.65, 0.93)	0.69 (0.50, 0.85)	0.91 (0.67, 0.98)	0.78 (0.56, 0.90)
Other algorithms when integrated with BiSAGE or OD						
GraphSAGE + OD	0.82 (0.70, 0.94)	0.97 (0.78, 0.99)	0.88 (0.75, 0.95)	0.85 (0.65, 0.92)	0.70 (0.61, 0.85)	0.78 (0.64, 0.88)
Autoencoder + OD	0.97 (0.79, 1.00)	0.88 (0.67, 0.92)	0.92 (0.72, 0.95)	0.52 (0.27, 0.70)	0.82 (0.61, 0.91)	0.64 (0.41, 0.78)
MDS + OD	0.98 (0.82, 1.00)	0.83 (0.63, 0.94)	0.90 (0.73, 0.93)	0.43 (0.15, 0.68)	0.87 (0.62, 0.96)	0.58 (0.32, 0.72)
BiSAGE + Feature bagging	0.91 (0.78, 0.95)	0.85 (0.78, 0.93)	0.87 (0.80, 0.93)	0.85 (0.72, 0.90)	0.93 (0.84, 0.98)	0.89 (0.79, 0.94)
BiSAGE + iForeast	0.86 (0.70, 0.91)	0.97 (0.89, 1.00)	0.90 (0.79, 0.93)	0.94 (0.86, 1.00)	0.80 (0.68, 0.87)	0.86 (0.78, 0.91)
BiSAGE + LOF	0.89 (0.75, 0.95)	0.89 (0.76, 0.96)	0.89 (0.74, 0.95)	0.88 (0.72, 0.90)	0.83 (0.68, 0.92)	0.85 (0.72, 0.90)

A. Overall Performance

We show overall comparison results of GEM against other state-of-the-art algorithms in Table I. GEM achieves the *best* performance in all the metrics. This is because the bipartite graph modeling of RF signal records and BiSAGE have well-learned similarities among signal records. Moreover, our outlier detection algorithm is also more robust to statistical fluctuations in outlier score prediction. Thus, outliers can be easily identified. SignatureHome has relatively low precision and recall in outside detection, indicating that it may have problems in separating signals observed near the boundary of the house since its network-based approach is not able to capture any perimeter information. INOA suffers from low precision in outside detection, which means that the support vectors generated from subsets of MACs do not represent 'inside' signals well and in turn lead to lower recall for in-premises detection. Furthermore, as mentioned in Section III-A, both SignatureHome and INOA have the missing-value problem, where a matrix representation of RF signals requires missing entries to be filled with some small values (i.e., -120 dBm), thereby degrading their performance.

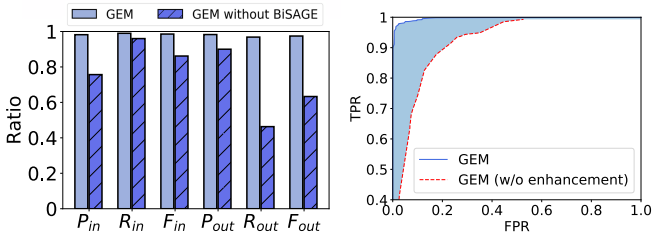
Effectiveness of BiSAGE: To study the effectiveness of our bipartite graph modeling and BiSAGE in GEM, we show the performance comparison of GEM, GraphSAGE, autoencoder, and MDS in Table I. For autoencoder and MDS, we construct a matrix of RF signal records, with missing entries filled with -120 dBm. As shown in Table I, GEM exhibits superior performance over the other algorithms (24%–67% in F_{out}). This verifies that the embeddings learned from BiSAGE are much more accurate in preserving the similarities and differences among signal records. On the other hand, GraphSAGE treats the bipartite graph as a homogeneous graph. It does not take into account node heterogeneity in the aggregation process, and its resulting embeddings are less accurate in representing the observed samples and their relationships. Autoencoder has low recall for in-premises detection, indicating that quite a few in-premises records are misclassified as outliers, i.e., P_{out} is low. This implies that the missing-value problem that arises when using a matrix for RF signals results in inaccurate inference of the similarities among signal records. A similar issue happens with MDS.

To see the gains in GEM owing to the bipartite graph modeling and BiSAGE, we show the performance comparison results of GEM with and without using the embeddings by BiSAGE in Figure 7(a). For GEM without the embeddings, we construct a matrix of signal records and fill in empty entries with -120 dBm. As seen from Figure 7(a), GEM improves the performance significantly (around 14% in F_{in} and 54% in F_{out}) when using embeddings as input, since BiSAGE is able to effectively learn the similarities among RF signals.

Effectiveness of our outlier detection algorithm: We use the same embeddings learned from BiSAGE as input for different outlier detection algorithms, namely feature bagging, iForest, and LOF, and compare their performance with GEM in Table I. As can be seen from Table I, GEM has the highest F-scores in both in-premises detection and outside detection, with *further* improvements of up to 12% in F_{in} and 18% in F_{out}. This is because our enhanced outlier detection algorithm 'stabilizes' statistical fluctuations and achieves better in-out classification performance along with new samples over time. iForest also has high recall for in-premises detection. However, its precision is not as good as that of GEM, which indicates that iForest is not able to filter out some outside signals properly. LOF does not show satisfactory performance in outside detection as it may not be able to classify the signals around the house boundary. Feature bagging may encounter a similar issue in outlier detection.

To see the effectiveness of our enhancement in the histogram-based outlier detection, we plot the receiver operating characteristic (ROC) curves [59] for GEM with and without our enhancement in Figure 7(b).[4] Here we focus on in-premises detection. For the ROC curve of a model, the closer to the upper left corner (a perfect classifier), i.e., the larger area under the ROC curve, the better the model is. As shown in Figure 7(b), with the same true positive rate, GEM achieves a much lower false-positive rate, implying that it greatly reduces the chance of missing the detection of outliers. We also observe that recall of outlier detection can *hardly* be improved by the *original* histogram-based outlier detection algorithm. This indicates that a non-trivial portion of the signal records obtained outside the geofencing area are consistently misclassified. On the contrary, our enhancement significantly improves the outlier detection performance by

[4]The ROC curve plots true positive rate vs. false positive rate (or false alarm rate) with *varying* threshold values, so it shows the overall performance of a classification model across different threshold values.



(a) (b)

Fig. 7. (a) Performance comparison of GEM and GEM without the embeddings by BiSAGE; (b) Performance comparison of GEM with and without our proposed enhancement in the histogram-based outlier detection algorithm.

TABLE II
USER-LEVEL PERFORMANCE OF GEM.

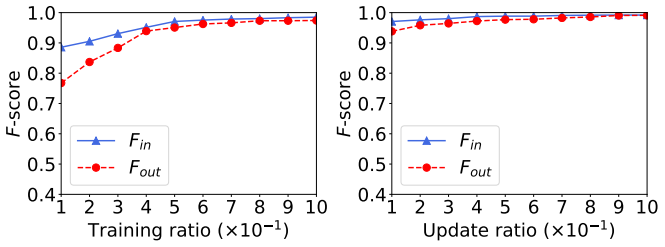
User	In-premises detection			Outside detection			#MACs	Area
	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	F_{out}		(m^2)
1	1.00	0.94	0.97	0.97	1.00	0.99	20	$\sim$10
2	0.94	1.00	0.97	1.00	0.97	0.98	26	$\sim$10
3	1.00	0.99	0.99	0.99	1.00	0.99	33	$\sim$50
4	0.98	1.00	0.99	1.00	0.88	0.94	16	$\sim$50
5	0.96	1.00	0.98	1.00	0.88	0.94	20	$\sim$50
6	0.98	1.00	0.99	1.00	0.94	0.97	65	$\sim$100
7	1.00	0.97	0.98	0.88	1.00	0.94	45	$\sim$100
8	1.00	1.00	1.00	1.00	1.00	1.00	73	$\sim$100
9	1.00	0.98	0.97	1.00	0.98	0.99	57	$\sim$100
10	0.94	1.00	0.97	1.00	0.98	0.99	12	$\sim$200
Avg.	0.981	0.989	0.985	0.982	0.968	0.974	36.7	$\sim$77

clearly separating the two groups of normal and abnormal signal records.

Benefits of model training and its online update: To see how many samples are needed for training our outlier detection model, we divide the (initial) training samples into ten equal sets and train GEM with one more set each time to perform outlier detection. As shown in Figure 8(a), the performance gradually improves with more training samples available, and GEM can *even* work with only 10% of training samples (less than 50 records on average), indicating its practicability in real deployment. A key design of GEM is that it leverages new incoming samples to improve the in-out classification accuracy over time. To evaluate this self-enhancement, we divide the 'testing' data into ten equal sets and present step-wise model update results in Figure 8(b). The performance of GEM improves with new samples streaming in, showing that it leverages highly confident in-premises samples to improve the detection accuracy.

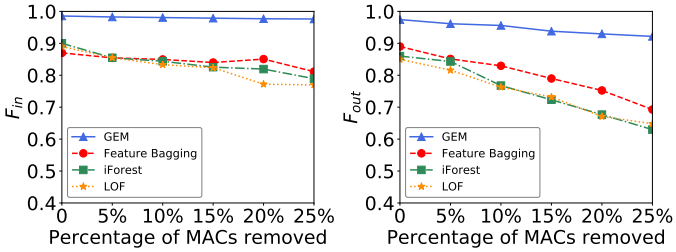
B. User-level Performance

We show quantitative results for each user in Table II. As seen from Table II, GEM performs well (with most F-scores higher than 0.95) across various housing types with different numbers of MACs seen. For multiplex apartments (≤ 100 m^2), more MACs lead to higher accuracy. On the other hand, for a two-story house (~ 200m^2), which is 'detached' from others, although the number of MACs is small, it should be easier to detect outside events. Thus, GEM remains effective for accurate geofencing. All the user studies validate the practicability of GEM.



(a) (b)

Fig. 8. (a) Performance vs. training ratio; (b) Performance vs. update ratio.



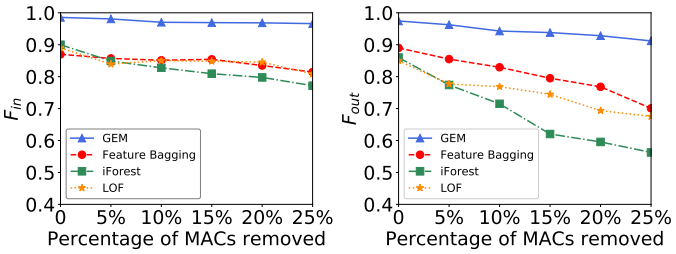
(a) In-premises detection (b) Outside detection

Fig. 9. Performance vs. MAC removal ratio in training set.

C. Micro-benchmark Evaluation

Adaptation to the changes in APs: To test GEM's adaptability to AP dynamics, we intentionally remove a subset of APs from the training and testing sets. Note that they are all ambient APs sensed, and we do not install any additional AP on site. In each experiment, we randomly remove up to 25% of MACs in total and repeat 30 runs to measure the average performance. Figure 9 shows F-scores by pruning MACs in the training set while leaving the testing set untouched. While F-scores of GEM decrease slowly as more MACs are removed, it remains significantly better than the other algorithms. As shown in Figure 9, the performance of GEM for in-premises detection almost remains intact, while the performance for outside detection decreases only slightly. It is because GEM keeps updating the outlier detection model with the highly confident in-premises samples. In other words, any newly sensed MACs can naturally be added into our bipartite graph, and they actually improve the in-out detection performance over time. Therefore, the removal of MACs in the training set does not significantly affect the performance. In addition, we intentionally remove MACs in the testing set, while leaving the training set unchanged. Similar results are observed, as shown in Figure 10.

Robustness to RF dynamics: To further evaluate the robustness of GEM to dynamic RF environments, we conduct an 'AP ON-OFF' experiment where some RF records associated with each AP/MAC remain intact ('ON') or disappear ('OFF') in the training and testing sets, and the (dis)appearance is governed by a two-state Markov model, as depicted in Figure 11. For each (p, q) pair, each AP/MAC stays in its current state with probability $1-p$ (resp. $1-q$) or moves to the other state with p (resp. q), and each state transition, including self-transition, takes place every 30 samples throughout the training and testing sets. This experiment is repeated 30 times for



(a) In-premises detection (b) Outside detection

Fig. 10. Performance vs. MAC removal ratio in test set.

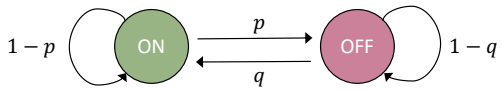


Fig. 11. Two-state Markov model for APs.

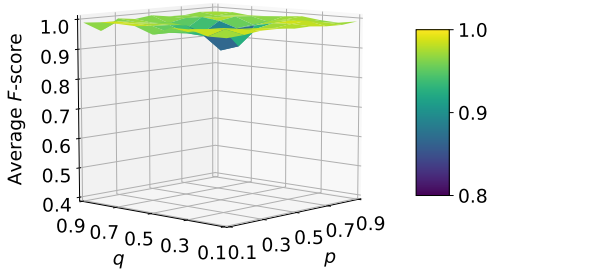
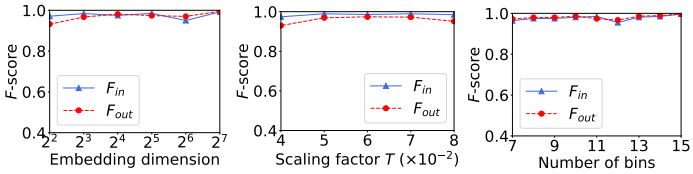


Fig. 12. Performance of GEM is robust to signal dynamics.



(a) (b) (c)

Fig. 13. Impact of (a) embedding dimension d, (b) scaling factor T, (c) bin size m, on the performance of GEM.

in-premise and outside detections, and the average F score is reported in Figure 12. Here we vary the values of p and q from 0.1 to 0.9. As shown in Figure 12, GEM performs well regardless of the choices of p and q, demonstrating its robustness to dynamic RF environments. It is worth noting that when (p, q) gets close to (0.5, 0.5), the *entropy rate* of its corresponding two-state Markov model increases, i.e., the Markov model has *higher uncertainty*. The higher uncertainty of the model contributes to a small dip in the average F score around $(p, q) = (0.5, 0.5)$. See [60] for more details on the entropy rate of Markov chains.

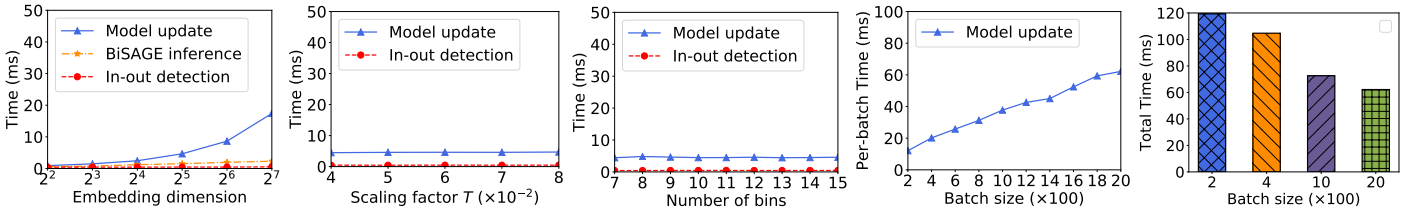
Tolerance to parameter perturbation: From the bipartite graph, GEM learns node embeddings via BiSAGE and uses the embeddings for outlier detection. Thus, it is important to check how the choice of embedding dimension would affect the system performance. In Figure 13(a), we show the F-scores when the embedding dimension varies. We can see that GEM exhibits outstanding performance regardless of the choices of the dimension, implying its robustness to such changes.

Similarly, in Figure 13(b), we present the overall system performance of GEM when the scaling factor T in Equation (11) changes. We can see that GEM is not sensitive to the choices of T from 0.04 to 0.08 and achieves very high F scores overall. Thus, we can easily tune the scaling factor T to achieve highly accurate performance in practice. In addition, Figure 13(c) presents the effect of the number of bins m on the performance in outlier detection. Recall that our enhanced histogram-based outlier detection algorithm divides the range of values along each dimension into m bins and builds d histograms for outlier detection. Our algorithm achieves excellent performance over a wide range of values for m.

Inference time measurement: It is also important to see if GEM is able to make the in-out decision *in a prompt manner* upon arrival of a new RF signal record. To this end, we measure how long the entire inference process takes and obtain the breakdown of the inference time. Recall that the inference process has three steps, namely (1) obtaining the embeddings of the new RF record using BiSAGE, (2) in-out detection by our outlier detection algorithm, and (3) online model update with the new record. All results here are obtained by taking the average from 2000 runs.

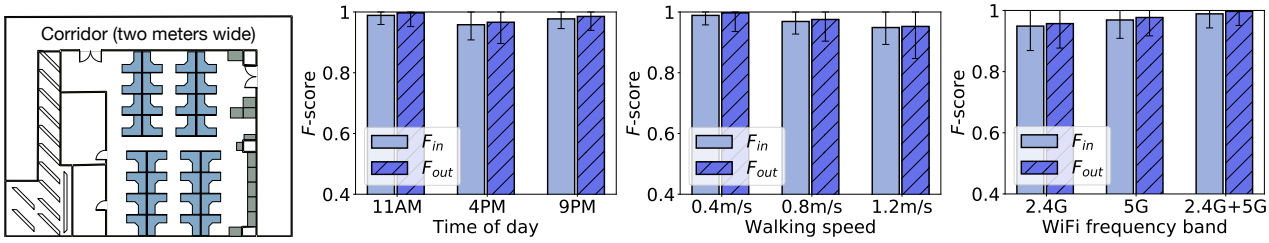
We first show the impact of embedding dimension d on the inference time in Figure 14(a). Since the embedding dimension affects all three steps in the inference process, we provide their running times. As shown in Figure 14(a), the embedding dimension has little impact on the running times of BiSAGE inference and in-out detection, while the running time of model update increases with d. It is because the model update involves updating d histograms. A higher dimension indicates more histograms to update, thereby leading to a longer time for the update. Nonetheless, the total inference time is still less than 20 milliseconds even when $d = 128$. In addition, we evaluate the impact of scaling factor T and the number of bins m on the inference time. Since they only affect the inference process (i.e., in-out detection and model update) after the embeddings are learned by BiSAGE, we only report the running times of in-out detection and model update in Figure 14(b) and Figure 14(c). We see that T and m have little impact on the inference time. This again confirms the computationally inexpensive inference process of GEM, which is suitable for real-time in-out detection.

We further discuss the impact of batch size on the inference time when we update our outlier detection model in batch mode. Recall that our model is updated only with *highly confident* signal records. Since the batch update only affects the model update in the inference process, we only report its running time in Figure 14(d) and Figure 14(e). We first evaluate how long it takes to update the model with *one* batch of (highly confident) signal records. As shown in Figure 14(d), the running time increases with the batch size since it takes longer time to recompute histograms by including a larger number of new records. We also evaluate the running time of updating the model using a total of 2000 signal records with different batch sizes. For example, we have ten batches with the batch size of 200, while having two batches with the



(a) (b) (c) (d) (e)

Fig. 14. Running time versus (a) embedding dimension, (b) scaling factor T, and (c) bin size m; (d) Running time of processing a batch with different sizes; (e) Running time of processing 2000 samples with different batch sizes, e.g., 10 batches need to be processed for the batch size of 200.



(a) Lab floorplan (b) Different time instances (c) Walking speed variations (d) Different frequency bands

Fig. 15. Lab experiments.

batch size of 1000. As shown in Figure 14(e), the running time of model update decreases with increasing batch size as a reduction in the number of batches outweighs a saving in the model-update time with a smaller batch size. Thus, in practice, we can resort to the model update in batch mode.

TABLE III
RSS VARIATION DURING A DAY.

Time	Mean (dBm)	SD (dBm)	#MACs
11 AM	−67.76	8.63	68
4 PM	−81.67	13.50	85
9 PM	−78.64	11.03	47

D. Impact of Environmental Factors

We carry out further experiments to study the impact of environmental factors, such as time-varying RSS values, the variation in the walking speed of each user for collecting initial training data, and less availability of WiFi frequency bands, on the performance of GEM. Each experiment is conducted once a day for three consecutive days in a lab environment (see Figure 15(a)). Note that the testing data for each experiment covers the area inside the lab (the geofencing area) and the corridor outside, which is only two meters wide, i.e. the boundary space of the geofencing area.

RSS variations over time. RF signal strength varies at different times of the day, mainly due to the people and their devices in the area. We select three representative time instances, i.e., 11AM, 4PM, and 9PM to test GEM's performance. At 11AM and 4PM, many people walk around the lab, and thus the signal strength may vary significantly. While at 9PM, signals would be more stable as the environments get quieter. Table III summarizes the statistics of RSS values measured during a day, where SD stands for standard deviation. We collect the initial training data at 11AM and then conduct 50 walks around the lab for testing GEM at each time instance. Thanks to the model updating mechanism, GEM can adapt to time-varying RSS values and thus achieve outstanding performance consistently, as shown in Figure 15(b). In other words, GEM remains effective even in such a dynamic RF signal environment.

Walking speed variations: For initial training, each user is asked to walk around the house for a short period of time. By varying the user's walking speed, we carry out three different experiments, i.e., slow walk ($\sim$0.4m/s), normal walk ($\sim$ 0.8m/s), and fast walk ($\sim$ 1.2m/s). For each experiment, we walk along the inner perimeter of the lab twice for initial training. For testing purposes, we conduct 50 walks around (inside and outside) the lab. As shown in Figure 15(c), GEM exhibits excellent performance in all three cases.

Different frequency-band availabilities: We study the geofencing performance versus APs' frequency bands as more devices support dual and tri-bands. Here, we train and test the model with signals of 2.4GHz only, 5GHz only, and 2.4GHz + 5GHz. As presented in Figure 15(d), more frequency bands (i.e., 2.4GHz + 5GHz) improve the performance, even though our system remains effective in the 2.4GHz only environment. We also observe that a higher frequency band results in better performance, as higher frequency signals are more likely confined within the lab space.

E. Model Scalability

To validate the scalability of our model with large-scale datasets, we conduct additional experiments in a five-story shopping mall (Figure 16) and based on the UJI open dataset [61] from Kaggle competition, which covers three buildings and contains 21,048 RF signal records. For each building, we define the middle floor as the geofenced area and the other floors as outside. In the shopping-mall experiment, we walk around the third floor (middle floor) to collect about 5,000 signal records for initial training. We then walk randomly within the five-story building to collect about 200,000 signal records for testing. For the experiment on the UJI

TABLE IV
PERFORMANCE COMPARISON IN THE SHOPPING MALL DATASET.

Algorithms	In-premises detection			Outside detection		
	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	F_{out}
GEM	**0.94**	**0.99**	**0.96**	**0.98**	**0.96**	**0.97**
SignatureHome	0.68	0.85	0.75	0.76	0.72	0.74
INOA	0.75	0.89	0.81	0.80	0.78	0.79

TABLE V
PERFORMANCE COMPARISON IN BUILDING 0 OF THE UJI OPEN DATASET.

Algorithms	In-premises detection			Outside detection		
	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	F_{out}
GEM	**0.91**	**0.99**	**0.95**	**0.99**	**0.98**	**0.99**
SignatureHome	0.62	0.87	0.72	0.85	0.83	0.84
INOA	0.68	0.89	0.77	0.88	0.85	0.86

TABLE VI
PERFORMANCE COMPARISON IN BUILDING 1 OF THE UJI OPEN DATASET.

Algorithms	In-premises detection			Outside detection		
	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	F_{out}
GEM	**0.86**	**0.99**	**0.92**	**0.99**	**0.97**	**0.98**
SignatureHome	0.56	0.84	0.67	0.79	0.78	0.78
INOA	0.65	0.88	0.75	0.86	0.82	0.84

TABLE VII
PERFORMANCE COMPARISON IN BUILDING 2 OF THE UJI OPEN DATASET.

Algorithms	In-premises detection			Outside detection		
	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	F_{out}
GEM	**0.84**	**0.99**	**0.91**	**0.99**	**0.98**	**0.99**
SignatureHome	0.50	0.82	0.62	0.82	0.77	0.79
INOA	0.59	0.85	0.69	0.85	0.82	0.83

dataset (per building), we uniformly sample half of the signal records from the middle floor (i.e., around 800 records) for initial training and use the rest of the records for testing.

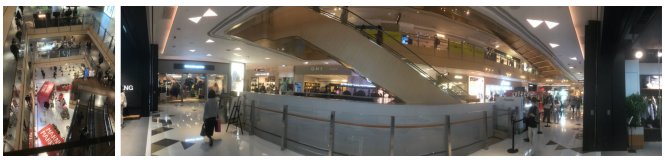


Fig. 16. Shopping mall layout.

We compare GEM with SignatureHome and INOA and report the results in Tables IV–VII. GEM outperforms SignatureHome and INOA significantly. It is because the embeddings learned by BiSAGE well preserve the relative proximity between signal records, and our enhanced outlier detection algorithm accurately predicts outliers from normal data samples. However, SignatureHome does not benefit from an associated IP within the geofencing area as the experiments are done based only on RF signals. Moreover, since signals from an AP/MAC can be detected on multiple floors, a mere calculation of the overlap ratio of MACs in SignatureHome can hardly differentiate the signal records collected within the geofencing area form the ones outside. In addition, although INOA performs better than SignatureHome, it also suffers from the same problem. The nature of RF signals from an AP/MAC that can reach different floors makes the learning from the signal records for each pair of APs/MACs insufficient for accurate in-out detection.

VI. DISCUSSION

One may be concerned about the vulnerability of our online model update scheme to some potential attack by 'bad actors' who intentionally stay at the boundary and move outward slowly to abuse the online model update and thus break the system. However, this is *hardly* possible in practice. In a typical geofencing scenario such as nursing home or medical observation, the boundary of the premises is usually well defined and clear, i.e., the geofenced region and outside are often separated by walls or partitions. Such physical objects bounce RF signals off and attenuate the signals, e.g., a signal attenuation of 3 dB for drywalls (half signal strength) and up to 10 dB for brick walls (10 times less signal strength), thereby making the signal characteristics inside and outside quite different. In other words, the physical objects not only obstruct the bad actors in staying at the boundary but also prevent them from building up a collection of RF signals whose strengths are gradually changing. In addition, GEM only leverages highly confident samples (around 5% of newly predicted in-premises samples) to update the 'signal boundary'. The signals collected outside the boundary are quite unlikely to be used for our online model update.

VII. CONCLUSION

We have proposed GEM, a practical yet effective geofencing system that only leverages ambient RF signals without any extra hardware or continuous intervention from users. GEM is built upon three integral components, i.e., a representation of RF signals via a weighted bipartite graph, our novel bipartite network embedding algorithm BiSAGE, and our enhanced histogram-based detection algorithm. They enable GEM to achieve highly accurate in-out detection performance while being robust to dynamic RF environments. We have empirically demonstrated the robustness and superior performance of GEM in various housing and RF scenarios.

We expect that the key enablers of GEM have broader impacts. The bipartite graph modeling and BiSAGE could be used as a viable solution to dealing with *variable-length* feature vectors for building a learning model. BiSAGE could also be applied for network embedding and representation learning on *general* bipartite graphs with other applications.

REFERENCES

[1] J. Helmy and A. Helmy, "The alzimio app for dementia, autism amp; alzheimer's: Using novel activity recognition algorithms and geofencing," in *IEEE SMARTCOMP*, 2016.

[2] J. Tan, E. Sumpena, W. Zhuo, Z. Zhao, M. Liu, and S.-H. G. Chan, "Iot geofencing for covid-19 home quarantine enforcement," *IEEE Internet of Things Magazine*, vol. 3, no. 3, pp. 24–29, 2020.

[3] E. Hermand, T. W. Nguyen, M. Hosseinzadeh, and E. Garone, "Constrained control of uavs in geofencing applications," in *IEEE MED*, 2018.

[4] R. R. Oliveira, I. M. Cardoso, J. L. Barbosa, C. A. da Costa, and M. P. Prado, "An intelligent model for logistics management based on geofencing algorithms and rfid technology," *Expert Syst. Appl.*, 2015.

[5] A. Greenwald, G. Hampel, C. Phadke, and V. Poosala, "An economically viable solution to geofencing for mass-market applications," *Bell Labs Tech. J.*, 2011.

[6] H. Zang, F. Baccelli, and J. Bolot, "Bayesian inference for localization in cellular networks," in *IEEE INFOCOM*, 2010.

[7] S. Nirjon, J. Liu, G. DeJean, B. Priyantha, Y. Jin, and T. Hart, "Coin-gps: Indoor localization from direct gps receiving," in *MobiSys*, 2014.

[8] J. Schloemann, H. S. Dhillon, and R. M. Buehrer, "Toward a tractable analysis of localization fundamentals in cellular networks," *IEEE Trans. Wirel. Commun.*, 2015.

[9] H. Rizk, M. Torki, and M. Youssef, "Cellindeep: Robust and accurate cellular-based indoor localization via deep learning," *IEEE Sens. J.*, 2018.

[10] R. Margolies, R. Becker, S. Byers, S. Deb, R. Jana, S. Urbanek, and C. Volinsky, "Can you find me now? evaluation of network-based localization in a 4g lte network," in *IEEE INFOCOM*, 2017.

[11] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent neural networks for accurate rssi indoor localization," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 639–10 651, 2019.

[12] L. Li, X. Guo, and N. Ansari, "Smartloc: Smart wireless indoor localization empowered by machine learning," *IEEE Trans. Ind. Electron.*, 2019.

[13] R. Wang, H. Luo, Q. Wang, Z. Li, F. Zhao, and J. Huang, "A spatial–temporal positioning algorithm using residual network and lstm," *IEEE Trans Instrum Meas*, 2020.

[14] M. Zhou, Y. Li, M. J. Tahir, X. Geng, Y. Wang, and W. He, "Integrated statistical test of signal distributions and access point contributions for wi-fi indoor localization," *IEEE Trans. Veh. Technol.*, 2021.

[15] S. Fan, Y. Wu, C. Han, and X. Wang, "Siabr: A structured intra-attention bidirectional recurrent deep learning method for ultra-accurate terahertz indoor localization," *IEEE J. Sel. Areas Commun.*, 2021.

[16] X. Chen, H. Li, C. Zhou, X. Liu, D. Wu, and G. Dudek, "Fidora: Robust wifi-based indoor localization via unsupervised domain adaptation," *IEEE Internet Things J.*, 2022.

[17] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," in *KI-2012*, 2012.

[18] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, 2004.

[19] R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint*, 2018.

[20] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *ICML*, 2018.

[21] S. Goyal, A. Raghunathan, M. Jain, H. V. Simhadri, and P. Jain, "Drocc: Deep robust one-class classification," in *ICML*, 2020.

[22] M. Z. Zaheer, J.-H. Lee, M. Astrid, and S.-I. Lee, "Old is gold: Redefining the adversarially learned one-class classifier training paradigm," in *IEEE CVPR*, 2020.

[23] P. Liznerski, L. Ruff, R. A. Vandermeulen, B. J. Franks, M. Kloft, and K.-R. Müller, "Explainable deep one-class classification," in *ICLR*, 2021.

[24] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.

[25] K.-H. Chow, S. He, J. Tan, and S.-H. G. Chan, "Efficient locality classification for indoor fingerprint-based systems," *IEEE Trans. Mobile Comput.*, 2019.

[26] M. Boysen, C. de Haas, H. Lu, X. Xie, and A. Pilvinyte, "Constructing indoor navigation systems from digital building information," in *2014 IEEE 30th International Conference on Data Engineering (ICDE)*. IEEE, 2014.

[27] H. Li, H. Lu, X. Chen, G. Chen, K. Chen, and L. Shou, "Vita: A versatile toolkit for generating indoor mobility data for real-world buildings," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1453–1456, 2016.

[28] A. I. Baba, M. Jaeger, H. Lu, T. B. Pedersen, W.-S. Ku, and X. Xie, "Learning-based cleansing for indoor RFID data," in *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, 2016.

[29] Y. Lin and D. Jiang, "LOCATER: Cleaning WiFi Connectivity Datasets for Semantic Localization," *Proceedings of the VLDB Endowment*, 2021.

[30] M. Gao, L. Chen, X. He, and A. Zhou, "Bine: Bipartite network embedding," in *ACM SIGIR*, 2018.

[31] W. Zhuo, Z. Zhao, K. H. Chiu, S. Li, S. Ha, C.-H. Lee, and S.-H. G. Chan, "GRAFICS: Graph Embedding-based Floor Identification Using Crowdsourced RF Signals," in *ICDCS*, 2022.

[32] J. You, X. Ma, Y. Ding, M. J. Kochenderfer, and J. Leskovec, "Handling missing data with graph representation learning," in *NeurIPS*, 2020.

[33] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *ACM KDD*, 2017.

[34] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2018.

[35] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020.

[36] S. Chatzopoulos, K. Patroumpas, A. Zeakis, T. Vergoulis, and D. Skoutas, "SPHINX: A system for metapath-based entity exploration in heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2913–2916, 2020.

[37] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 6, pp. 854–867, 2020.

[38] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou, "Dynamic Heterogeneous Information Network Embedding With Meta-Path Based Proximity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 3, pp. 1117–1132, 2022.

[39] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, and Q. Wang, "Interpretable and efficient heterogeneous graph convolutional network," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[40] T. Gu, C. Wang, C. Wu, Y. Lou, J. Xu, C. Wang, K. Xu, C. Ye, and Y. Song, "HybridGNN: Learning Hybrid Representation for Recommendation in Multiplex Heterogeneous Networks," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022.

[41] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019.

[42] C. Yang, A. Pal, A. Zhai, N. Pancha, J. Han, C. Rosenberg, and J. Leskovec, "Multisage: Empowering gcn with contextualized multi-embeddings on web-scale multipartite networks," in *ACM KDD*, 2020.

[43] C. Huang, Y. Fang, X. Lin, X. Cao, W. Zhang, and M. Orlowska, "Estimating Node Importance Values in Heterogeneous Information Networks," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022.

[44] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *ACM KDD*, 2017.

[45] M. Corain, P. Garza, and A. Asudeh, "DBSCOUT: A density-based method for scalable outlier detection in very large datasets," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021.

[46] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, "Robust and explainable autoencoders for time series outlier detection," in *Proceeding of the 38th IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2022.

[47] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef, "WiDeep: WiFi-based accurate and robust indoor localization system using deep learning," in *IEEE PerCom*, 2019.

[48] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting," *Big Data Anal.*, vol. 3, no. 1, pp. 1–17, 2018.

[49] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *ACM KDD*, 2014.

[50] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013.

[51] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *J. Mach. Learn. Res.*, vol. 9, no. 11, 2008.
[52] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop*, 2013.
[53] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *IEEE CVPR*, 2020.
[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[55] M. A. A. Cox and T. F. Cox, "Multidimensional scaling," in *Handbook of Data Visualization*. Springer, 2008, pp. 315–347.
[56] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *ACM KDD*, 2005.
[57] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *IEEE ICDM*, 2008.
[58] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *ACM SIGMOD*, 2000.
[59] M. H. Zweig and G. Campbell, "Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine," *Clin. Chem.*, 1993.
[60] M. Thomas and A. T. Joy, *Elements of information theory*. Wiley-Interscience, 2006.
[61] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems," in *2014 international conference on indoor positioning and indoor navigation (IPIN)*. IEEE, 2014.