Satellite Navigation
https://satellite-navigation.springeropen.com/

ORIGINAL ARTICLE

Open Access

Accurate and efficient floor localization with scalable spiking graph neural networks

Fuqiang Gu[1], Fangming Guo[1], Fangwen Yu[2], Xianlei Long[1*], Chao Chen[1], Kai Liu[1], Xuke Hu[3], Jianga Shang[4] and Songtao Guo[1]

Abstract

Floor localization is crucial for various applications such as emergency response and rescue, indoor positioning, and recommender systems. The existing floor localization systems have many drawbacks, like low accuracy, poor scalability, and high computational costs. In this paper, we first frame the problem of floor localization as one of learning node embeddings to predict the floor label of a subgraph. Then, we introduce FloorLocator, a deep learning-based method for floor localization that integrates efficient spiking neural networks with powerful graph neural networks. This approach offers high accuracy, easy scalability to new buildings, and computational efficiency. Experimental results on using several public datasets demonstrate that FloorLocator outperforms state-of-the-art methods. Notably, in building B0, FloorLocator achieved recognition accuracy of 95.9%, exceeding state-of-the-art methods by at least 10%. In building B1, it reached an accuracy of 82.1%, surpassing the latest methods by at least 4%. These results indicate FloorLocator's superiority in multi-floor building environment localization.

Keywords Indoor positioning, Deep learning, Floor localization, Spiking neural networks, Graph neural networks

Introduction

Indoor positioning has become increasingly popular because of its widespread applications. It determines the location of a target using positioning signals such as WiFi (Zhou et al., 2022), Bluetooth (Zhao et al., 2023), inertial sensors (Gu et al., 2018b), vision (Zhao et al., 2023), and light (Zhuang et al., 2018). So far, plenty of indoor positioning systems have been proposed and developed, yet most of them have focused on achieving 2D positioning. In complex structures like multi-floor buildings, the task of floor identification is paramount. Floor localization is a fundamental basis for plenty of applications and services such as emergency response and rescue (Weinlich et al., 2018), indoor positioning (Gu et al., 2019b; El-Sheimy & Li, 2021), and recommender systems (Deldjoo et al., 2020).

Floor identification methods can be categorized as *fingerprinting* (Zhang et al., 2020) and *sensor-based methods* (Qi et al., 2019; Ye et al., 2016). The fingerprinting approaches, including WiFi fingerprinting and cellular fingerprinting, are very popular due to the wide availability of WiFi and cellular infrastructures. The typical fingerprinting systems for floor localization include *SkyLoc* (Varshavsky et al., 2007), *StoryTeller* (Elbakly & Youssef, 2020) and *ZeeFI*(Gu et al., 2019a). Yet, such methods need a large amount of training data, which increases with the number of floors as well as the area of interest. Besides, the data need frequently re-collected to keep the fingerprints update, which results in the poor scalability of these methods.

To expedite the site surveys of these classical fingerprinting approaches, many sensor-based methods have

*Correspondence:
Xianlei Long
xianlei.long@cqu.edu.cn
[1] College of Computer Science, Chongqing University, Chongqing, China
[2] Department of Precision Instrument, Tsinghua University, Beijing, China
[3] Institute of Data Science, German Aerospace Center (DLR), Jena, Germany
[4] School of Computer Science, China University of Geosciences, Wuhan, China

been proposed, which can reduce the amount of training data by using the inertial sensors and/or barometers. Popular sensor-based systems include *FTrack* (Ye et al., 2012), *F-Loc* (Ye et al., 2014), *B-Loc* (Ye et al., 2016) and *BarFi* (Shen et al., 2015), etc. While sensor-based methods can reduce the time and effort for site surveys by using additional sensors such as inertial sensors and barometers, they suffer from a limited coverage since these sensors are not available in all devices.

Recently, deep neural networks have been successfully applied in various domains, such as natural language processing (Vaswani et al., 2017), image classification (Krizhevsky et al., 2012), activity recognition (Gu et al., 2018a), and indoor positioning (Wang et al., 2016). Because deep learning models have the advantages of more powerful representational ability, better generalizability, and unnecessary feature engineering, they have also been applied to floor identification. *StoryTeller* (Elbakly & Youssef, 2020) uses Convolutional Neural Networks (CNNs) for floor identification, where WiFi signals are first used to generate images that are then fed to a CNN for predicting floors. While *StoryTeller* is both Access Point (AP)-independent and building-independent, it requires the knowledge of physical building dimensions and 3D locations of APs, which are impractical in some scenarios. *ZeeFI* (Gu et al., 2019a) utilizes stacked autoencoders to identify floors, alleviating the effort for data collection by automatically recognizing the ground floor with smartphone-built sensors. However, it uses only two layers of autoencoders, and hence its representational ability is limited and might not well deal with more complex cases. In (Zhang et al., 2020), a cellular-based floor identification method is introduced, which first uses a denoising autoencoder for data noise reduction and feature extraction and then utilizes a Long Short-Term Memory (LSTM) network for floor identification. However, these methods still suffer from the limitations including poor scalability, low accuracy, high computational cost, and requirement for additional information (e.g., building dimensions, locations of APs).

The motivation of this study is to develop a novel floor localization method that is scalable, accurate, robust, and computationally efficient. Existing methods often require a predefined and regular structure, which is not always feasible given the dynamic nature of WiFi APs. In this study, we introduce FloorLocator - a deep learning-based floor identification method that integrates computation-efficient Spiking Neural Networks (SNNs) with powerful Graph Neural Networks (GNNs). By organizing APs into a graph structure, we can effectively handle situations where the exact positions of AP nodes are unknown. This approach offers a more adaptive and robust solution compared to conventional methods. Additionally, the integration of SNNs into the system is driven by their high computational efficiency, making them an optimal choice for real-time floor localization tasks.

The basic idea of *FloorLocator* is illustrated in Fig. 1. We first represent raw WiFi scans into a RSS vector of values between 0 and 1 using the powed method (Torres-Sospedra et al., 2015). Then, we transform the RSS vector to a graph of visible APs before using a Spiking Graph Neural Network (SGNN) to learn the mapping between the 'WiFi graph' and the actual floor label. It includes two phases: offline training and online testing. In the offline phase, each RSS vector in the fingerprint database is first organized in a WiFi fingerprint graph, which is then fed into the SGNN together with its corresponding floor label. After that, the network is trained by minimizing the loss between the predicted label and the actual label. In the online phase, the upcoming WiFi scan is first transformed into a RSS vector, which is then expressed as a WiFi fingerprint graph. The trained network takes the fingerprint graph as input and predicts the floor label of the input WiFi scan.

In this work, FloorLocator is designed to offer several advantages over existing SOTA floor localization methods, including *StoryTeller* (Elbakly & Youssef, 2020). Specifically, based on graph theory, FloorLocator innovatively utilizes graph topology to learn representations, which enhances the transferability of the model. Then, to achieve computational efficiency and energy savings, FloorLocator employs an event-driven SNN architecture. Finally, we perform extensive experiments to verify *Floorlocator*'s robustness.

- We propose FloorLocator: a novel deep learning-based floor identification method, which reduces the burden of conventional WiFi fingerprinting by using SNN and GNN.
- *FloorLocator* is both **AP-independent** and **building-independent**. *FloorLocator* is graph-based, and hence it can be easily applied to new buildings or environments by simply modifying the input graph.
- *FloorLocator* is **efficient** and **robust**. Its spiking-based operation allows it to function in an energy-efficient way. Also, it can integrate information over time and filter out irrelevant information, making it robust to noise.
- We implement and evaluate FloorLocator using publicly available datasets collected in three different buildings. Experimental results show that FloorLocator outperforms SOTA methods for floor identification. Our method has higher accuracy, better scalability, and greater efficiency compared to existing methods.

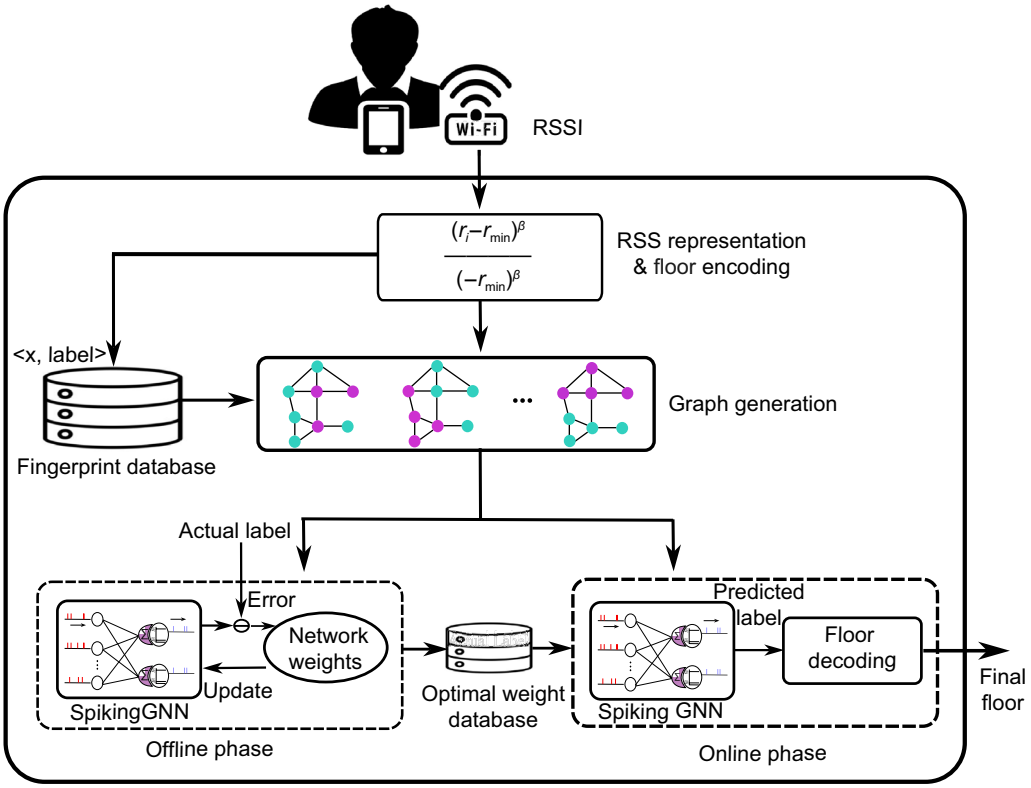


Fig. 1 Overview of *FloorLocator*. It takes as input WiFi scans, which are organized in a graph of visible APs before feeding into a spiking graph neural network for training and predicting. Each module of *FloorLocator* will be detailed in Section V

To the best of our knowledge, FloorLocator is the first work that integrates event-driven SNNs with GNNs for floor identification.

Method and system

In this section, we first introduce the theoretical foundation of the proposed method, including GNNs and SNNs. Then, we present the problem formulation of floor identification with SGNN before elaborating on the proposed method.

Neural networks

This section describes the foundational concepts of neural networks, with a focus on GNNs and SNNs, which form the core of the proposed FloorLocator system. We illustrate how GNNs, renowned for their effectiveness in learning complex structures within graph data, manage the intricate relationships between architectural elements to facilitate precise floor localization. Concurrently, we introduce the dynamics of SNNs, highlighting their computational efficiency and their bio-inspired mechanisms that mimic human neural activity patterns.

Graph neural networks

GNNs, a subset of neural networks, excel in feature extraction through node interactions in graphs. Their applications cover various fields, notably in graph mining (Li et al., 2019), object classification (Gu et al., 2020), recommender systems (He et al., 2020), and antibiotic discovery (Stokes et al., 2020). This study particularly focuses on Graph Convolutional Networks (GCNs), which leverage spectral domain convolutions through adaptable graph filters (Bianchi et al., 2021).

Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be a graph, where $\boldsymbol{V} = \{v_1, \cdots, v_n\}$ is the set of nodes, $\boldsymbol{E} = \{e_1, \cdots, e_m\}$ is the set of edges, and $e_k = e_{ij}$ denotes an edge pointing from v_i to v_j. The adjacency matrix $\boldsymbol{A}$ of the graph is an $n \times n$ matrix, and $A_{ij} = 1$ if $e_{ij} \in \boldsymbol{E}$, otherwise, $A_{ij} = 0$. Thus, the node embedding $\boldsymbol{H}^{(l+1)}$ of a GCN layer in (Kipf & Welling, 2017) is described as:

$$\boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{1}$$

where σ is an activation function (e.g., ReLU), $\boldsymbol{W}^{(l)}$ is a layer-specific learnable weight matrix, $\tilde{\boldsymbol{A}}$ is the normalized adjacency matrix of $\boldsymbol{A}$, and

$$\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-1/2} \tag{2}$$

where $\boldsymbol{I}$ is the identity matrix.

Early GCN works often need to compute the spectrum of the graph Laplacian (Bruna et al., 2014) or approximate the spectrum (Defferrard et al., 2016) using high-degree Chebyshev polynomials of the Laplacian matrix, which is computationally expensive. In contrast, the GCN approach in (Kipf & Welling, 2017) simplifies this by using first-order Chebyshev polynomials of the graph Laplacian to cut down computational costs. Further streamlining is seen in (Du et al., 2018), which employs adjacency matrix polynomials up to the second degree, further reducing complexity. This efficient and effective method is utilized in our current work.

Spiking neural networks

SNNs are brain-inspired neural networks that inherit the biological spatial-temporal dynamics mechanisms and rich spiking coding schemes (Roy et al., 2019). SNNs, being closer to biological neural mechanisms than DNNs, are very suitable for neuroscience-inspired models and are compatible with energy-efficient neuromorphic hardware like Intel Loihi and Tianjic. Key neuron models in SNNs are the Spike Response Model (SRM) (Gerstner, 1995) and Leaky Integrate-and-Fire (LIF) model (Wu et al., 2018).

We first introduce the SRM (Shrestha & Orchard, 2018). where an input spike train $x_i(t)$ enters a neuron, incorporating the refractory kernel ν nd the neuron's output spike train $s(t)$. In this model, input spikes are converted into spike response signals $a_i(t)$. These are scaled by synaptic weights w_i to produce post-synaptic potentials. Consequently, the neuron's membrane potential, $u(t)$, is determined by summing these potentials and refractory responses, which is written as:

$$u(t) = \boldsymbol{w}^T \boldsymbol{a}(t) + (\nu * s)(t) \tag{3}$$

An output spike is fired when the membrane potential surpasses a pre-defined threshold. The spike function $f_s(\cdot)$ can be written as:

$$f_s(u) : u \to s, s(t) := s(t) + \delta(t - t^{(f+1)}) \tag{4}$$

where

$$t^{(f+1)} = \min\{\mathrm{t} : \mathrm{u(t)} = \mathrm{u_T}, \mathrm{t} > \mathrm{t^{(f)}}\} \tag{5}$$

Another popular SNN model is the LIF model, which is more computationally tractable than SRM models while maintaining biological fidelity to some extent.

The dynamics of LIF is governed by:

$$\tau \frac{\mathrm{du(t)}}{\mathrm{dt}} = -u(t) + \sum_i w_i x_i \tag{6}$$

where $u(t)$ represents the internal membrane potential of a neuron at time t, $\sum_i w_i x_i$ is the weighted summation of the inputs from pre-neurons, and τ is a time constant. Figure 2 visualizes the computational model of a SNN. In this study, we use the LIF model due to its higher accuracy, lower computational cost, and easer training.

Problem formulation

We formulate the problem of floor identification with SGNN as a problem of finding node embeddings to predict labels of a subgraph (corresponding to a WiFi fingerprint) given a graph with node attributes. Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ denote the fingerprint graph, where a node $v_i \in \boldsymbol{V}$ is an Access Point (AP), and an edge $e_{ij} \in \boldsymbol{E}$ is the edge connecting two APs v_i and v_j that are spatially close to each other, which means that the two APs should at least appear once in a fingerprint vector of the radio map. The learning process of node embedding for floor identification includes two steps:

(i) *Aggregating messages* The task of this step is to aggregate messages from neighboring APs, which is written as:

$$\boldsymbol{m}_v^{(l)} = f^{(l)}\left(\boldsymbol{h}_v^{(l)}, \{\boldsymbol{h}_\zeta^{(l)} : \zeta \in \mathcal{N}(v)\}\right) \tag{7}$$

where $\boldsymbol{h}_v^{(l)}$ and $\boldsymbol{m}_v^{(l)}$ denote the node embedding and the message vector of AP v at l-th layer, $f^{(l)}$ represents the aggregation function, $\mathcal{N}(v)$ is the neighboring nodes of v.

(ii) *Transforming messages* The task of this step is to transform messages to the next layer. Mathematically, the process is described as:

$$\boldsymbol{h}_v^{l+1} = g^{(l)}(\boldsymbol{m}_v^{(l)}) \tag{8}$$

where $g^{(l)}$ is the transformation function at l-th layer. For batch execution, the above equation can be written as:

$$\boldsymbol{H}^{(l+1)} = f_{LIF}(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{9}$$

where f_{LIF} is the LIF activation function, and other symbols are the same as described in Eq. (1).

Then, the objective of floor identification with SGNNs is to train the model by minimizing the total loss L which includes the supervised loss and graph regularization term, namely

$$L = L_0 + \lambda L_{\mathrm{reg}} \tag{10}$$

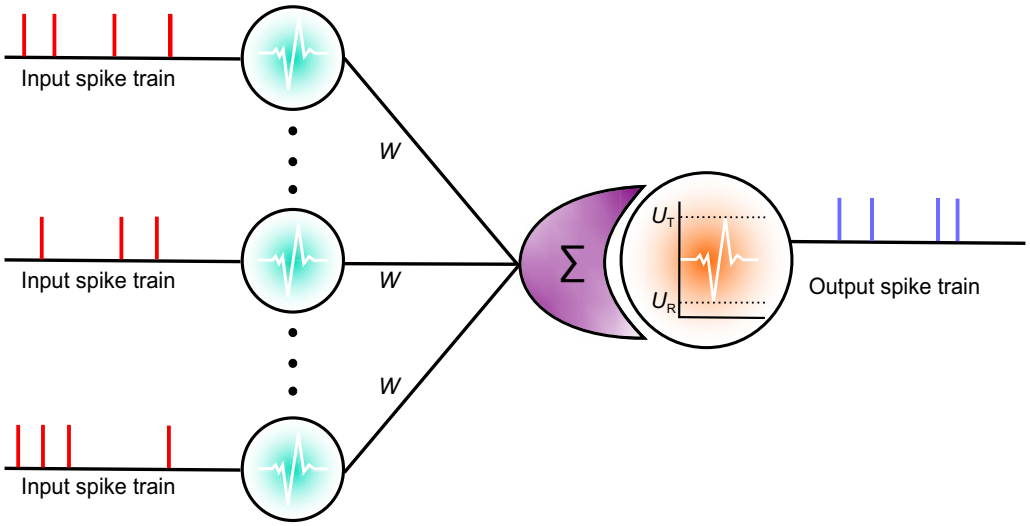


Fig. 2 SNN computational model. It consists of a post-neuron driven by input pre-neurons. The membrane potential of the post-neuron is affected by the input spikes from pre-neurons

where L_0 is the supervised loss, λ is a weighting factor and L_{reg} is the graph regularization term. In our work, we use the mean squared error as the supervised loss, namely

$$L_0 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{11}$$

where y_i is the ground truth floor label for the i-th fingerprint, and $\hat{y}_i$ is the estimated floor label. The graph regularization term can be written as:

$$L_{\text{reg}} = \sum_{ij} \boldsymbol{A}_{ij}\|\varphi(\boldsymbol{X}_i) - \varphi(\boldsymbol{X}_j)\|^2 \tag{12}$$

where $\varphi(\cdot)$ is a GNN-like differentiable function, and $\boldsymbol{X}_i$ is the node feature vector for AP v_i.

The *FloorLocator* system

In this section, we first describe the architecture of *FloorLocator*, followed by an introduction of RSS representation & floor encoding, and WiFi graph generation. Then we elaborate on each of its components: TAGConv Layer, LIF module, DeepBlock module, Event-based Batch Normalization module, FC Layer & Voting Layer module. Finally, we introduce network training and floor decoding.

System architecture

The FloorLocator system is specifically designed to address the challenges of floor localization mentioned in the introduction section. The architecture of FloorLocator is illustrated in Fig. 3. The system accepts WiFi fingerprints (which can also be cellular fingerprints) as input. These fingerprints are initially represented as RSS vectors and subsequently organized into a graph based on the proximity of their APs. This graph-based representation is particularly beneficial in scenarios where the positions of the AP nodes are unknown, allowing for a flexible and adaptive structure. Following this, the graph data undergoes a transformation into spikes after the first Topology

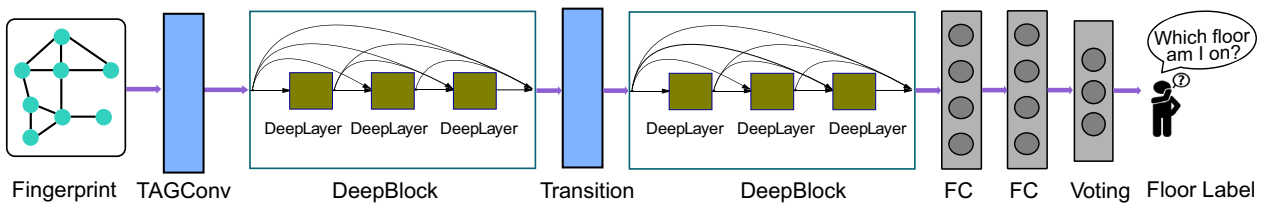


Fig. 3 System architecture of *FloorLocator*. The input of *FloorLocator* is a WiFi fingerprint, and its output is the estimated floor label for the given fingerprint. It consists of one TAGConv layer, two DeepBlocks, one transition layer (which is also a TAGConv layer), two FC layers and one voting layer. Each DeepBlock is composed of three DeepLayers, and these DeepLayers are densely connected. Note that each TAGConv layer is followed by an LIF activation and event-based batch normalization layer, while each FC layer is followed only by an LIF activation. These subsequent layers are not shown in the figure for clarity

Adaptive Graph Convolutional (TAGConv) (Du et al., 2018) layer using the LIF activation function. The spike-converted data is then processed.

RSS representation and Floor encoding

In this section, we first introduce the fingerprint data representation method. The number of visible APs changes with location, and hence the size of each fingerprint vector may be different. To fix the size of the fingerprint vector, we describe a fingerprint as a vector of RSS from all the APs in the environment. Let $\boldsymbol{x}$ indicate the fingerprint, and $\boldsymbol{x} =< r_1, r_2, \cdots, r_M >$, where r_i represents the signal strength received from the i-th AP and M is the number of APs in the environment. To better learn features from fingerprints via SGNNs, we describe the fingerprint in positive values using the powed representation method in (Torres-Sospedra et al., 2015). The raw RSS rss_i is then described as a positive value p_i as follows:

$$f_i = \begin{cases} \frac{(r_i - r_{\min})^\beta}{(-r_{\min})^\beta}, & r_i \geq \tau \\ 0, & r_i < \tau \end{cases} \tag{13}$$

where τ is a RSS threshold (we set min as the threshold), indicating if an is detected in a fingerprint. β is constant parameter, which is simply set to the mathematical constant e. These APs with RSS lower than τ are considered as not-detected. Thus, the fingerprint $\boldsymbol{x}$ can be re-written as a vector of positive values of all the APs in the environment, namely

$$\boldsymbol{x} =< p_1, p_2, \cdots, p_M > \tag{14}$$

Then, we describe how the floor information is encoded into an identity matrix for easier processing. Floor labels are generally categorical or are not fully numeric and need to be converted to numbers (mostly integers) before fed into a DNN or SNN. In this work, we encode the floor label into an identity matrix with one-hot encoding, where each row indicating a floor, has and only has one element with value 1 representing the floor.

Graph generation

Classical fingerprinting methods need to store the APs' IDs, which limit their scalability. A recently developed method called *StoryTeller* ((Elbakly & Youssef, 2020) alleviates this requirement by transforming fingerprints into images. While *StoryTeller* is scalable, it requires the locations of APs, which is impractical in some cases. In this study, we organize the fingerprints in a graph, which does not require the APs' IDs, neither the locations of APs. This makes our method more scalable and applicable to different environments.

Specifically, we generate the fingerprint graph according to the closeness of APs. Each visible AP is taken as a node v_i of the graph, and each edge e_{ij} connecting two nodes (APs) represents the two APs close to each other. When the two APs appear in the same fingerprint, we add an edge to connect them until all the fingerprints are traversed. The detailed steps of constructing the fingerprint graph are represented in Fig. 4. The algorithm takes as input a radio map D, and outputs the fingerprint graph. It first generates the number of APs N by computing the length of any fingerprint vector (e.g., $\boldsymbol{x}_0$). Then, we add each node into the node set of the graph. We use the index of an AP appearing in the fingerprint as the node ID, rather than their IDs. This allows our method to be scalable to new environments since it alleviates the dependence on APs' IDs, which exists in traditional fingerprinting methods. After that, we transverse each fingerprint, and extract the indexes of APs that are visible in the fingerprint into a set C. Finally, we add two edges $< v_i, v_j >$ and $< v_j, v_i >$ into the edge set E of the graph if

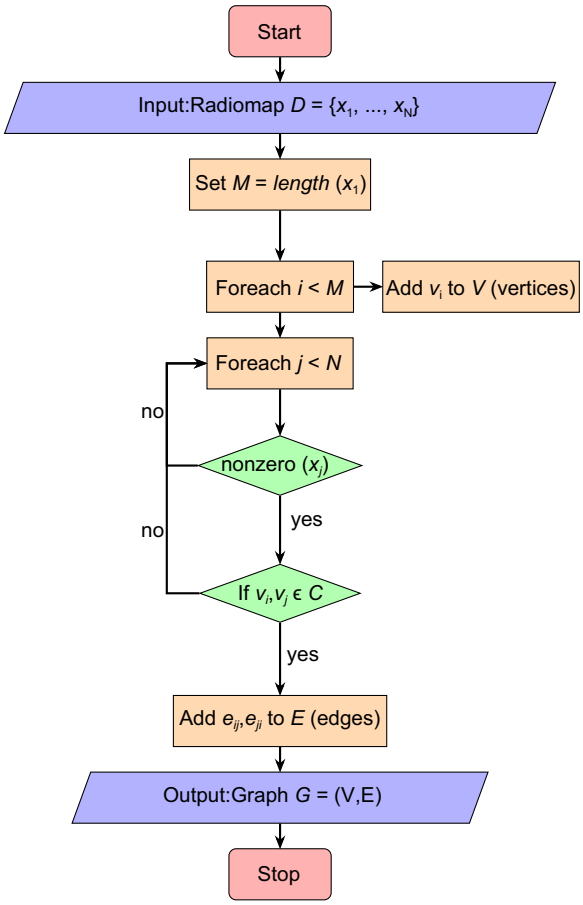


Fig. 4 Fingerprint Graph Generation

they are not in the edge set yet. The reason of adding the two edges is to construct an undirected fingerprint graph.

Details of FloorLocator

In this section, we delve into the intricate components that form the FloorLocator's architecture. The 'TAGConv Layer' adapts to topological nuances, a significant leap from conventional layers. Moving forward, the 'LIF Activation' implements the dynamics of spiking neurons, a departure from standard activation functions. At the heart of our architecture is 'DeepBlock,' a complex assembly that enhances the network's learning depth. Stability is a key in learning, and 'Event-based Batch Normalization' ensures this by mitigating the common gradient-related issues. Finally, the 'FC Layer and Voting Layer' work in unison to interpret and classify the processed data, a critical final step in our system's response mechanism.

TAGConv layer

Instead of using the popular graph convolution (Kipf & Welling, 2017), we adopt the TAGConv (Du et al., 2018) because of its excellent performance in terms of accuracy and computational efficiency. It utilizes a set of fixed-size learnable filters to simultaneously extract both node features and the strength of correlation between nodes. Let $\boldsymbol{G}_{c,k}$ represent the k-th graph filter. The resulting k-th feature map $\boldsymbol{h}_k^{(l)}$ on layer l is given by the equation:

$$\boldsymbol{h}_k^{(l)} = \sum_{c=1}^{C_l} \boldsymbol{G}_{c,k}^{(l)} \boldsymbol{x}_c^{(l)} + \boldsymbol{b}_k \tag{15}$$

where $\boldsymbol{x}_c$ is the c-th input feature vector of nodes, C_l denotes the count of input features per node at layer l, and $\boldsymbol{b}_k$ is a tunable bias vector. To adapt the convolution operation to various graph topologies, it's crucial to tailor the graph filter. A common method involves defining the graph filter using the normalized adjacency matrix $\tilde{\boldsymbol{A}}$ of the graph,

$$\boldsymbol{G}_{c,k}^{(l)} = \sum_{i=0}^{K} g_{c,k,i}^{(l)} \tilde{\boldsymbol{A}}^i \tag{16}$$

with $g_{c,f,k}$ as the graph filter's polynomial coefficient and $\tilde{\boldsymbol{A}}$ as the normalized adjacency matrix.

LIF activation

ReLU (Glorot et al., 2011) and its variants, like LReLU (Maas et al., 2013) are prevalent activation functions in CNNs. However, ReLU is not suitable for SNNs. Instead, we employ the LIF model, widely recognized for modeling spiking neuron dynamics (Roy et al., 2019). As depicted in Fig. 5, the LIF model illustrates that a neuron's membrane potential accumulates incoming spikes and experiences leakage over time. Upon reaching the threshold u_T, the neuron emits a spike and enters a refractory period.

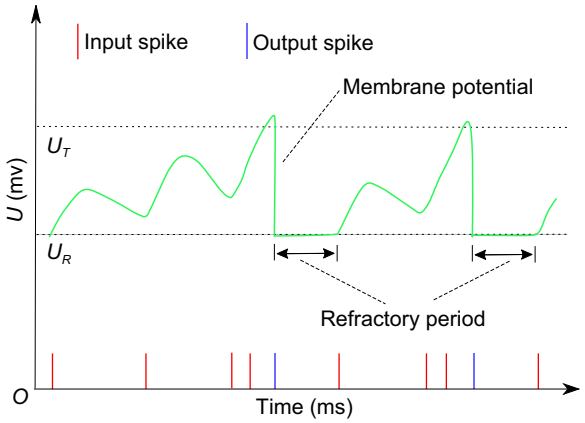


Fig. 5 The dynamics of LIF spiking neurons. The post-neuron integrates incoming spikes into its membrane potential, and fires a spike when the membrane potential surpasses a threshold u_R. After that, its membrane potential is set to a pre-set value u_R

The LIF activation function, f_l is mathematically expressed as:

$$f_l(u) = 1 \;\&\; u(t) \leftarrow u_R \;\; u(t) \geq u_T \tag{17}$$

where u_R and u_T denoting the reset value and firing threshold, respectively. Essentially, the LIF function triggers a neuron to fire a spike when its membrane potential reaches or exceeds u_T, following which the potential resets to u_R.

DeepBlock

DeepBlock is a key component of *FloorLocator*, which is composed of three DeepLayers. These DeepLayers are densely connected to learn features more effectively from graph-structured fingerprint data. Each DeepLayer contains two TAGConv layers, two event-based Batch Normalization (BN) layers, and two LIF activation layers. Such a design is inspired by the basic block of ResNet, which can effectively eliminate problems of gradient vanishing and explosion by adding skip connections.

Figure 6 compares the basic block of ResNet with the basic block (DeepLayer) of our design. The distinctions between ResNet and our proposed method are threefold:

- *Convolution Operation* Unlike the conventional convolution operation in ResNet, we employ TAGConv

to handle graph-structured fingerprint data. This choice is motivated by the capability of GNNs to effectively process WiFi APs without the prior knowledge of their structure. In contrast, CNNs organize nodes into a regular rectangular structure.

- *Activation Function* The commonly used ReLU function in ResNet is substituted with the LIF activation in our design. This change is essential because ReLU is incompatible with spiking data. SNNs are known for their computational efficiency, and their combination with GNNs can enhance the model's computational efficiency.
- *Normalization Layer* We replace the BN layer of ResNet with EBN to tackle gradient vanishing and explosion challenges.

Furthermore, DeepLayer is designed to take spikes as input and output spikes, diverging from the real-valued data processing in traditional networks. In the FloorLocator architecture, we incorporate two DeepBlocks. The number of DeepBlocks is determined by empirical study. Our experiments show that two DeepBlocks suffice to achieve excellent floor identification accuracy.

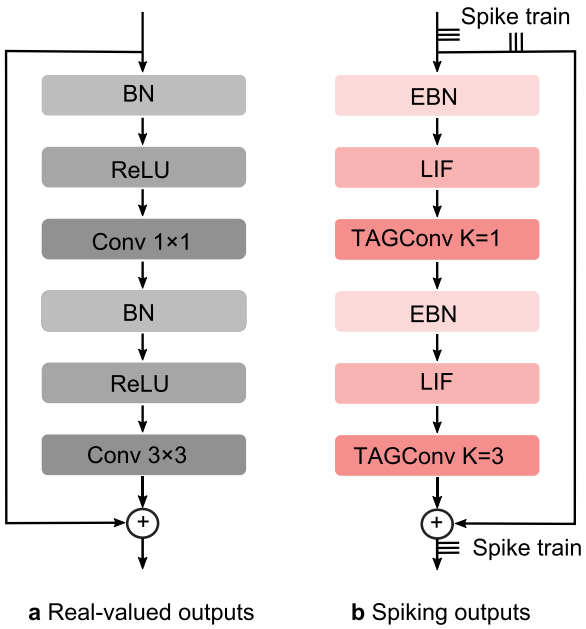


Fig. 6 Different basic blocks of commonly-used ResNet and the proposed method. **a** Basic block of ResNet. **b** Basic block (DeepLayer) of the proposed method

Event-based batch normalization

In this part, we introduce EBN in detail. As we know, gradient vanishing and explosion are the main problems that prohibit a model from going deeper. A common solution to the problems in deep neural networks is to utilize BN, which enables the model to converge stably and go deeper. However, BN cannot be directly used in SNNs due to the existence of additional temporal dimensions and different activation mechanisms. To avoid gradient vanishing and explosion in the proposed method, we adopt the EBN method (Zheng et al., 2021) and normalize the pre-synaptic inputs along the channel dimension. The inputs of each neuron are adjusted into the interval ranging from 0 to U_T, where U_T epresents the pre-defined spiking firing threshold (0.5 is used in this work). Such an adjustment can balance the inputs and neuronal membrane potential to avoid the membrane potential being over-saturated or the input information being over-expressed. Mathematically, the EBN is described as:

$$\hat{\boldsymbol{x}}_i^l = \frac{u_T(\boldsymbol{x}_i^l - E(\boldsymbol{x}_i^l))}{\sqrt{D(\boldsymbol{x}_i^l) + \epsilon}} \tag{18}$$

$$\boldsymbol{y}_i^l = \gamma_i \hat{\boldsymbol{x}}_i^l + \beta_i \tag{19}$$

where $\boldsymbol{x}_i^l$ represent the i-th channel feature map of $\boldsymbol{x}^l$, u_T represents the Spiking firing threshold, $E(\boldsymbol{x}_i^l)$ and $D(\boldsymbol{x}_i^l)$ denote the Expectation and Variance of $\boldsymbol{x}^l$ over the mini-batch. ϵ is a tiny constant to avoid dividing by zero error, $ee\gamma_i$ and β_i are two learnable parameters.

In the training process, the Expectation and Variance of $\boldsymbol{x}_i$ can be computing by

$$E(\boldsymbol{x}_i^l) = \text{mean}(\boldsymbol{x}_\text{i}^\text{l}) \tag{20}$$

$$D(\boldsymbol{x}_i^l) = \text{mean}((\boldsymbol{x}_\text{i} - \text{E}(\boldsymbol{x}_\text{i}^\text{l})^2) \tag{21}$$

In the inference stage, we cannot directly compute the Expectation and Variance of $\boldsymbol{x}_i$ due to the batch is not applicable. Therefore, we estimate the Expectation of $E(\boldsymbol{x}_i^l)$ and $D(\boldsymbol{x}_i^l)$ in the whole dataset, which can be obtained by moving average solution in the training stage. More details about such estimations can be found in (Zheng et al., 2021).

FC layer and voting layer

At the top of *FloorLocator* are two FC layers and one voting layer. The FC (Fully Connected) layer in our network operates similar to those in standard neural networks, defined as,

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{22}$$

where $\boldsymbol{x}$ is the inputs from previous layer, $\boldsymbol{W}$ is the weight matrix, $\boldsymbol{b}$ is the bias vector, and $\boldsymbol{h}$ is the output feature.

For decoding the network's output, we use a voting layer, following the approach in (Wu et al., 2019). Each output label is linked to a neuron in this layer. The class prediction is based on the neuron that receives the highest number of votes (or spikes) averaged across a time window.

Network training and online localization

Our network associates the output vector to each floor label by voting, and the final output $\boldsymbol{O}_L$ in a given time window is written as

$$\boldsymbol{O}_L = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{U}\boldsymbol{o}^t \tag{23}$$

where $\boldsymbol{U}$ represents the constant voting matrix used for decoding spikes, while $\boldsymbol{o}^t$ signifies the output feature from the final layer at time t.

To accommodate the error backpropagation, we take the mean square error between the average voting result and the label vector $\boldsymbol{y}$ as the loss function.

$$L = \left\|\boldsymbol{y} - \boldsymbol{O}_L\right\|^2 \tag{24}$$

However, the spiking function poses a significant challenge as it is not differentiable, which makes it impossible to use the error backpropagation method directly. To elaborate on this challenge, let's take a look at the expression for calculating weight gradients obtained through Spatial-Temporal Back-Propagation (STBP) (Wu et al., 2018):

$$\Delta W = \frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L}{\partial y(t)} \frac{\partial y(t)}{\partial u(t)} \frac{\partial u(t)}{\partial W} \tag{25}$$

In the above equation, y represents the target output vector, W denotes the SNN's weight matrix, $\mathcal{L}$ represents the loss function, and $\frac{\partial y(t)}{\partial u(t)}$ represents the gradient of the spiking activity function. The weight update formula is as follows:

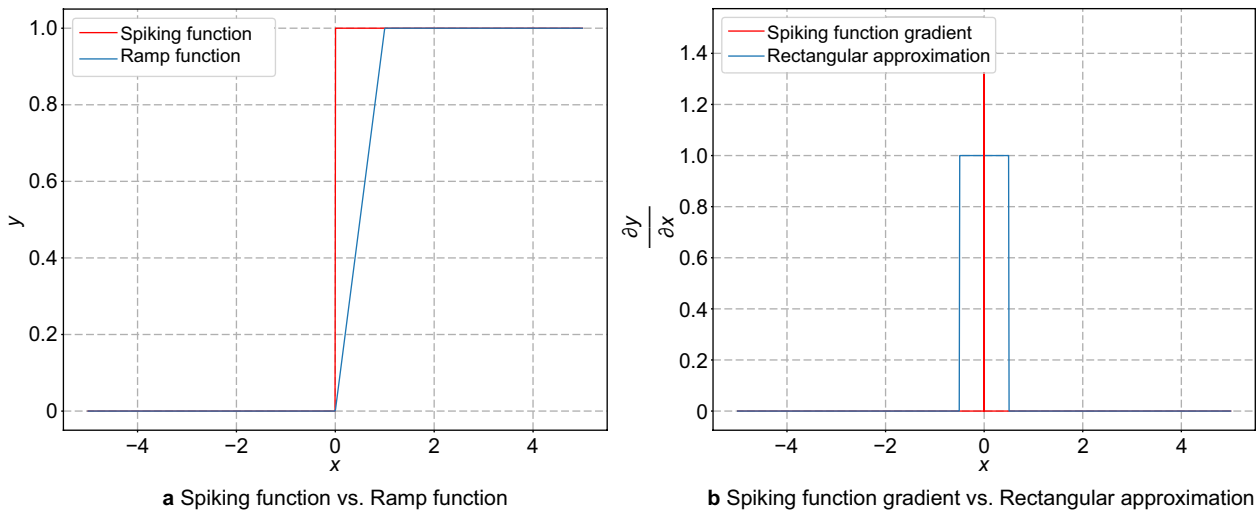
$$W = W - \eta \Delta W \tag{26}$$

Here, η represents the learning rate. However, because $\frac{\partial y(t)}{\partial u(t)}$ is either zero everywhere or a very large value in rare cases. As a result, the weights may not be updated at all, or they may be updated to a large value, leading to unstable training.

Previous studies tackled the non-differentiable challenge by using Surrogate Gradients (SG) (Wu et al., 2019; Shrestha & Orchard, 2018; Wu et al., 2018). To address this issue, we adopt a rectangular function (Wu et al., 2019) to approximate the derivative of the spiking function. Figure 7 illustrates the comparison between the rectangular function and the spiking activation function. The rectangular function is defined as follows:

$$f(u) = \frac{1}{a}\mathrm{sign}\left(|u - u_T| < \frac{a}{2}\right) \tag{27}$$

where a is a width parameter set to 0.5 in this study. In the ablation study, we will analyze the effect of different values.



a Spiking function vs. Ramp function

b Spiking function gradient vs. Rectangular approximation

Fig. 7 **a** Comparison of spiking and ramp activation functions: It can be observed that when the slope of the ramp function is sufficiently large, it approximates the spiking activation function; **b** Comparison of Spiking Activation Function Gradient and Surrogate Rectangular Function: When the width parameter is small, the figure of the rectangular function's gradient is similar to that of the spiking activation function

Table 1 Details of test scenarios

Building No.	Floors	Samples
B0	4	5785
B1	4	5503
B2	5	9760

Table 2 Hyperparameter setting

Parameter	Value
Optimizer	Adam
Learning rate	0.001
Number of epochs	100
Batch size	128
Spiking firing threshold	0.5
Width parameter *a*	0.5

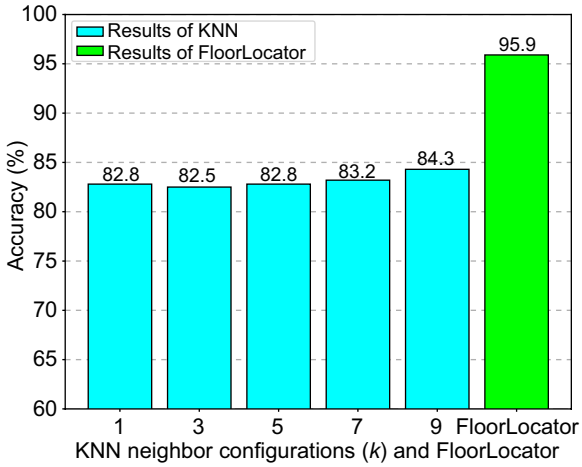


Fig. 8 Floor identification accuracy of *FloorLocator* as compared to the classical kNN in Building 0 (B0)

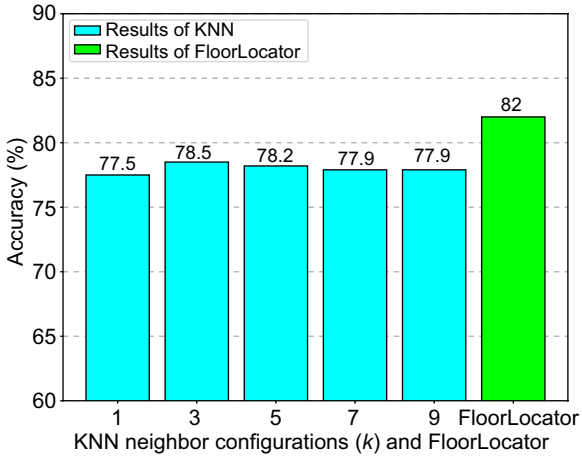


Fig. 9 Floor identification accuracy of *FloorLocator* as compared to the classical kNN in Building 1 (B1)

After training the network, it can predict the floor label of upcoming WiFi scans. The process of online floor localization is described below: first, the upcoming WiFi scan is represented as a RSS vector with values between 0 and 1, and then the RSS vector is expressed as a WiFi fingerprint graph according to the visible APs in the WiFi scan. After that, the WiFi graph is fed to the trained network to output the floor label, which is subsequently translated to a meaningful floor label (e.g., categorical character).

Experiments and results

Experimental setup

We evaluate the proposed method on three public scenarios (buildings) of the commonly used UJIIndoorLoc datasets (Torres-Sospedra et al., 2014).UJIIndoorLoc datasets are multi-building and multi-floor WiFi fingerprint datasets, which were collected with the assistance of more than 20 users using more than 25 Android devices. There are 520 APs visible in the database. Table 1 shows the details of the three test buildings.

While the dataset is relatively large, there still exists an imbalance in the data that the number of samples for each floor is not equal. Such an imbalance will affect the performance of deep learning models. To address this issue, we perform random oversampling to increase the number of samples in minority classes until a balance is reached. Such an operation is also for a fair comparison with SOTA *StoryTeller* (Elbakly & Youssef, 2020) that adopts the same strategy for data augmentation.

We implement the proposed method in PyTorch. The Adam optimizer is used to optimize the model on the training dataset. The number of epochs is set to 100, and the learning rate is 0.001. We repeated the training and test procedure for five rounds with different random seeds and reported the average accuracy. Table 2 shows the values of hyperparameters.

Experimental results

We first evaluate the proposed method on three test buildings and compare it with the classical

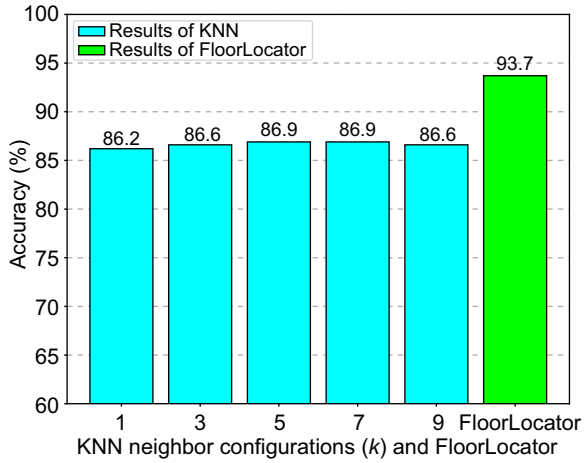


Fig. 10 Floor identification accuracy of *FloorLocator* as compared to the classical kNN in Building 2 (B2)

fingerprinting method - kNN (k Nearest Neighbors). The kNN is selected for performance comparison in ablation studies because of its popularity and widespread use in fingerprinting positioning. It often serves as a benchmark in many indoor localization systems and provides a baseline against which we can measure the improvement offered by the proposed FloorLocator. Later, we will also compare the performance of the proposed method with other SOTA methods. The value of k method ranges from 1 to 9 with an interval of 2. Figures 8, 9, 10 the floor identification accuracy in different buildings with the proposed *FloorLocator* compared to kNN method. It can be seen that *FloorLocator* significantly outperforms the popular in all three testbeds. The accuracy is improved by about 11.6% in Building 0, 3.5% in Building 1, and 6.8% in Building 2, respectively.

Compared to kNN, which needs to store APs' IDs and is building-dependent, our method is both AP-independent and building-independent, making it more scalable for new buildings. This attribute is due to the organization of fingerprints in a graph, which allows us to better capture the neighborhood correlation between samples on the same floor. These results indicate FloorLocator can learn useful feature embeddings from data, which helps the method capture more distinguishable features and get high accuracy. While kNN locates the floors using raw data, it results in a relatively lower accuracy.

Comparison to state-of-the-art methods

In this subsection, we compare the proposed method with SOTA methods for floor identification. The systems considered are as follows:

- *StoryTeller* (Elbakly & Youssef, 2020): A CNN-based method for floor identification. It first converts WiFi fingerprints into images that are fed to the VGG-19 (Simonyan & Zisserman, 2015) network.
- *kNN*: The popular method for floor localization, which is widely used (Luo et al., 2019).
- *TrueStory* (Elbakly et al., 2018): A neural network-based floor estimation method, which uses a multilayer perceptron network to fuse weak learners' outputs.
- *Locus* (Bhargava et al., 2015): A heuristics-based algorithm for floor identification.

Since the reference (Elbakly & Youssef, 2020) reported the performances of *StoryTeller*, *TrueStory*, and *Locus* in the Building 0 and 1 of the same UJIIndoorLoc dataset, we directly report their floor identification accuracy mentioned in (Elbakly &Youssef, 2020). For *kNN*, we implement it by ourselves. All methods are evaluated using the same training and test data for a fair comparison.

Figure 11 shows the performances of different methods in Building 0 and Building 1. It demonstrates that *FloorLocator* performs the best among these methods. Specifically, *FloorLocator* achieves about 96% and 82% correct floor estimates in Buildings 0 and 1, respectively. It surpasses *StoryTeller*, which is also AP-independent and building-independent, by about 16.5% in Building 0 and 19.3% in Building 1. The reasons why *FloorLocator* outperforms *StoryTeller* can be summarized as follows: first *StoryTeller* is a CNN-based approach, which needs to convert a WiFi scan to an image. During the conversion, there might be some information loss, affecting the performance of subsequent localization. By contrast, our method is graph-based, which uses the graph topology to learn useful representations. This allows it to better capture the spatial relationship between APs, which can be difficult to model using a CNN. Second, *FloorLocator* is more robust to noise than CNN-based methods. The integrate-and-fire neuron activation enables it to filter out irrelevant information.

It is also interesting to note that *kNN* performs better than StoryTeller and other baseline methods. This might lie in deep learning-based methods, including *StoryTeller* and *TrueStory*, which may lose some useful information in the representation and/or conversion of WiFi scans. However, *kNN* has poorer generalization ability to new environments compared to *FloorLocator* and *StoryTeller* that are easily scalable.

The above experiments of *FloorLocator* against *kNN* and other SOTA methods demonstrate the superiority of the proposed method, which not only extracts rich

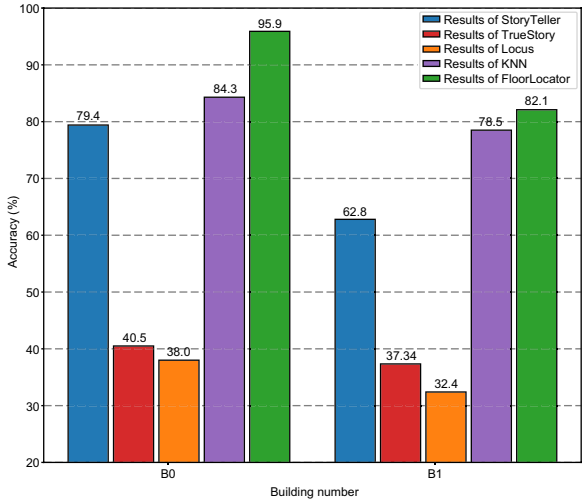


Fig. 11 Floor identification accuracy of *FloorLocator* (our method) as compared to other methods in Building 0 and Building 1

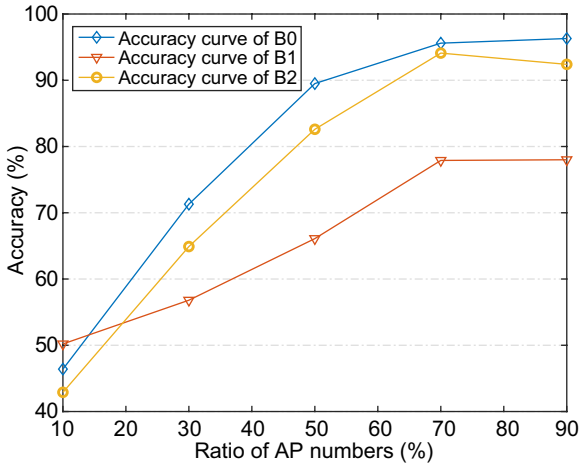


Fig. 12 Effect of AP density on the floor identification accuracy of *FloorLocator* in three test scenarios

floor-aware information, but also learns robust topology representations by filtering out useless features through the innovative SGNNs architecture.

Ablation experiment

To figure out the different impacts of different components involved in *FloorLocator*, we conducted several ablation studies. In the following, the effects of AP density, RSS threshold, RSS representations, width parameters, data size, and the generalization to new buildings will be presented and analyzed.

Effect of AP density

To understand the impact of AP density on *FloorLocator*'s performance, we conducted tests with varying AP densities. To do this, we randomly remove certain APs from both the training data and test data and use the remaining data for training the model and testing. AP density represents the ratio of the number of remaining APs to the total number of APs before the removal operation. Figure 12 shows how different numbers of APs in the training and test data affect the accuracy of the floor identification of our method. We can see that the accuracy decreases as the density of the APs used in the training declines. However, it is observed that FloorLocator can achieve relatively high accuracy even when 50% APs are removed, resulting in an accuracy of about 89.5% in Building 0, 66.1% in Building 1, and 82.6% in Building 2.

Figure 12 implies that *FloorLocator* is robust to the change of APs mainly due to the use of spike-based communication and activation mechanisms. *FloorLocator* can produce stable accuracy when there is a relatively high AP density, especially when AP density is greater than 70%.

Effect of RSS threshold

RSS threshold has an impact on the performance of the proposed method. Thresholding is a technique commonly used to remove irrelevant or noisy RSS values from fingerprints. To evaluate the effect of different threshold values on the performance, we conducted 100 training sessions on the B0 dataset and recorded the testing accuracy. The threshold values used range from to the minimum value in the dataset. Experimental results are shown in Fig. 13, indicating that a larger threshold value will filter out a substantial amount of information, leading to poorer performance. By contrast, a small threshold value will not reduce the testing accuracy yet remaining all values will somehow increase the computational complexity. Our analysis suggests that a threshold within the range of -85 to strikes an optimal balance. In this range, weaker, possibly irrelevant signals are effectively discarded, thereby optimizing computational efficiency without compromising accuracy. Notably, thresholds near appear to remove too many vital data, leading to a drop in performance.

Through the comparison in Fig. 13, we can draw that the selected range of -85 to -100dB ensures the retention of signals critical for achieving the desired testing accuracy, simultaneously filtering out the most disruptive noise.

Effect of RSS representations

We investigated the effect of various fingerprint representation methods on the performance of our model. Specifically, we considered four methods: positive values,

zero-to-one normalized values, exponential values, and powered values, as suggested by Torres et al. (Torres-Sospedra et al., 2015). After training our model for 100 epochs on the B0 dataset, we assessed the test accuracy. As shown in Fig. 14, our model effectively achieved optimal accuracy using the positive values, zero-to-one normalized values, and powered values representation methods. However, the exponential values method was less effective. A possible explanation for the underperformance of the exponential values method is its potential to produce the values greater than 1, some of which can be significantly high. This characteristic might be incompatible with the inherent features of our SNN. Specifically, SNNs emit spikes only under certain conditions within a specified time frame. If a normalization process provides unsuitable data, the neurons may become hyperactive, resulting in continuous firing. Given that the exponential method might yield a prevalence of high values, detecting subtle variations in the original dataset becomes increasingly challenging. Interestingly, the positive values representation seems to further hasten our method's convergence to optimal accuracy.

As can be seen in Fig. 14, the powed representation method can achieve a balance between accuracy and training efficiency.

Effect of width parameter

Since the spiking function is non-differentiable, we introduce a rectangle function to approximate the derivative of the spiking function. Here we analyze the influence of different width parameters of the rectangle function on the testing accuracy using the B0 dataset after training our model for 100 epochs. The value of the width parameter covers the commonly used values, namely 0.1, 0.5, 1.0, 2.0, 5.0, 7.5, and 10.0, respectively.

Figure 15 illustrates the relationship between our model's performance and the width parameter value, which ranges from 0.5 to 10.0. Within this range, our model tends to achieve optimal performance. In contrast, extremely low values, such as 0.1, impede the model's convergence on testing accuracy. Interestingly, while a smaller width parameter provides a more precise representation of the pulse function, the model's optimal performance appears to lean towards the larger values. This inclination might stem from the instances where the network grapples with significant gradient explosions, especially when the gradient closely resembles the pulse process. A broader rectangle function, offering smoother derivatives, aids the model in achieving a more seamless convergence. However, setting the width parameter too large, for instance beyond 10, leads to unpredictable training behaviors. This underlines the crucial task of judiciously selecting a width parameter. Such a choice should ideally encapsulate the dual objectives of aptly approximating the pulse function while also facilitating efficient model training.

Effect of data size

As we know, deep neural networks often require a large amount of data to train the models. Therefore, in this section, we evaluate the effect of data size on the performance of *FloorLocator*. Figure 16 demonstrates that *FloorLocator* is easier to train and achieves a good floor identification accuracy (about 94% in Building 0, 70.8% in Building 1, and 92.2% in Building 2) with only 30% This significantly reduces the time and effort required for collecting training data, allowing it to scale to new environments. Thus, *FloorLocator* can easily adapt to few-shot applications, which indicates the excellent feature

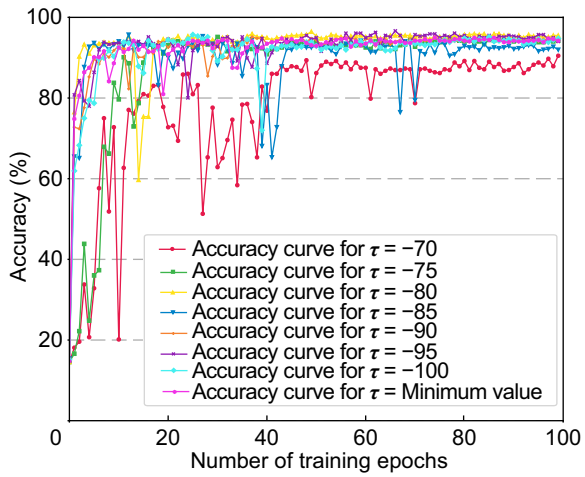


Fig. 13 Effect of RSS threshold (τ)

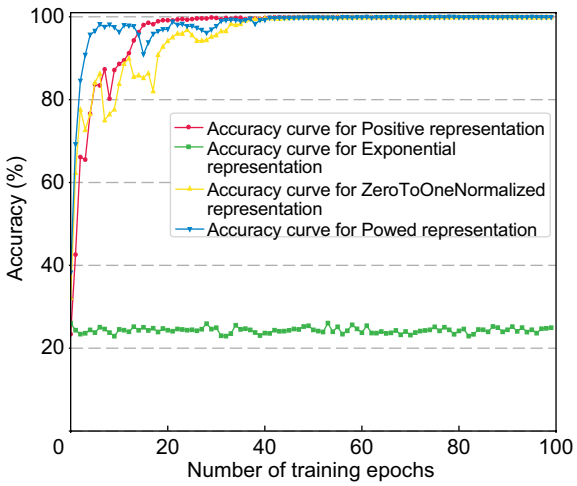


Fig. 14 Effect of RSS representation

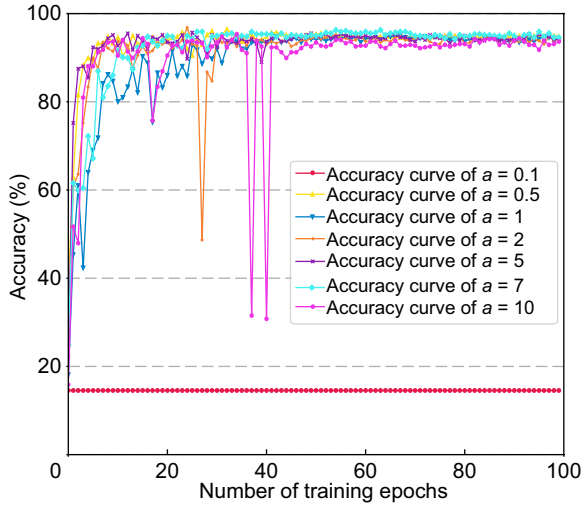


Fig. 15 Effect of width parameter (*a*) of the spiking derivative approximation function

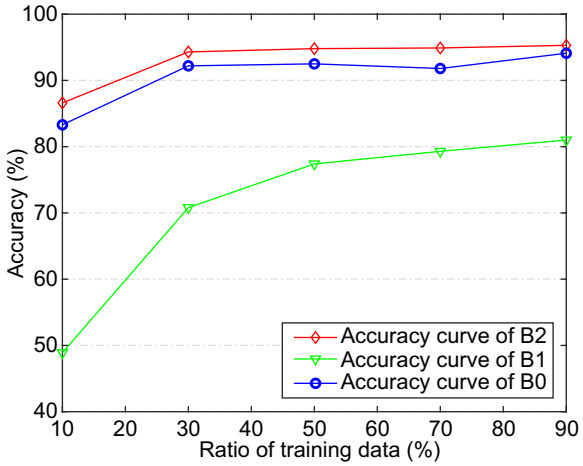


Fig. 16 Effect of the amount of used training data on the floor identification accuracy of *FloorLocator* in three test scenarios

extraction ability of FloorLocator within a limited number of samples.

Generalization to new buildings

Generalization ability is important for a floor identification system to be scalable. Generalization here means how well a model pre-trained in a building works in a new building with only a limited amount of training data from the new building. We trained our model on the training data from Building 0 and evaluated its performance in new buildings (Building 1 and Building 2) with different amounts of training data from the new buildings for fine-tuning the model.

Figure 17 demonstrates that about 10% new training data is enough to achieve floor identification accuracy close to the baseline where training and testing are conducted using the full data. The overhead of collecting such an amount of training data is negligible compared to classical fingerprinting methods that need to recollect a complete fingerprint database. This implies that the proposed method has excellent generalization ability and is scalable to new buildings. Such ability is attributed to the fact that we organize WiFi fingerprints in a graph of APs, making its easy adaptation to new scenarios by simply changing the input graph.

Through these ablation studies, we can conclude that *FloorLocator* has superior feature extraction ability by utilizing the data-oriented SGNN architecture. *FloorLocator* is not only robust to noisy signals but also shows good generalization ability across different buildings. Therefore, the effectiveness and scalability of the entire method are verified.

Conclusions and discussion

In this study, we proposed a novel floor identification method named *FloorLocator* utilizing SGNNs. This method is scalable, accurate, and efficient. Our findings show that FloorLocator notably surpasses both traditional fingerprinting techniques and contemporary deep learning methods in three test scenarios, demonstrating its robustness even in environments with complex architectural layouts and varied configurations.

However, *FloorLocator* still faces a challenge in large environments: the computational efficiency of *FloorLocator* will degrade dramatically when the number of in the environment is very huge. In such cases, the constructed input graph becomes a high dimensional tensor, which

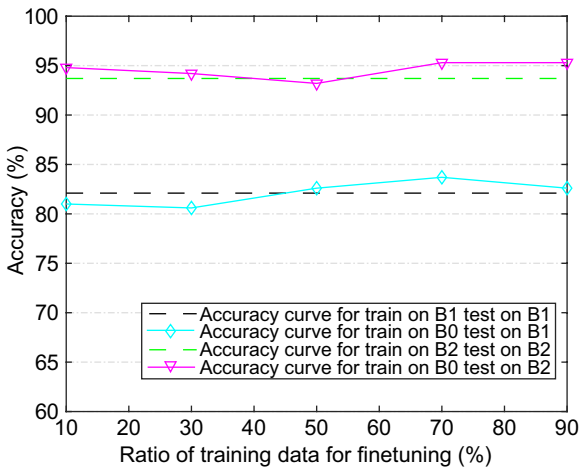


Fig. 17 Floor identification accuracy achieved using different amounts of data from a new building to fine-tune the model that is pretrained using some other building

makes the convolution operation very slow. This aspect is particularly critical in scenarios where real-time floor localization is required.

Despite these challenges, *FloorLocator* has demonstrated commendable resilience and accuracy in scenarios with sparse data available, much better than the conventional methods that rely heavily on dense WiFi data. This resilience highlights its potential for practical deployment in a variety of indoor settings, even those with limited access to comprehensive WiFi data.

In the future, we will extend *FloorLocator* to large environments by constructing the input graph in a more efficient way or dividing the environment into smaller areas so that the dimension of constructed input graphs is not too large. Optimizing the system for these extensive environments will require innovative approaches to data processing and neural network configuration, ensuring that *FloorLocator* maintains high-performance levels without compromising on the speed or accuracy.

Also, we will implement *FloorLocator* on neuromorphic hardware. The anticipated integration with neuromorphic hardware promises to further enhance *FloorLocator*'s computational efficiency, potentially transforming it into a more versatile and powerful tool for indoor floor localization. This advancement can extend the system to be used in real time scenarios such as emergency response, personalized indoor navigation, and location-based services.

In conclusion, *FloorLocator* stands as an evidence to the potential of SNNs in challenging real-world applications, setting a new benchmark in indoor floor localization. As we continue refining *FloorLocator*, we anticipate its evolution into a cornerstone technology in its domain, adaptable to the complexities of various indoor environments and capable of supporting a wide spectrum of applications.

Author contributions

Fuqiang Gu proposed the main idea, Fangming Guo conducted experiments, wrote up the manuscript under the supervision of Fuqiang Gu, and Chao Chen. Kai Liu, Songtao Guo, Xuke Hu, and Jianga Shang helped in the formulation of the main idea and experimental analysis. Fangwen Yu and Xianlei Long assisted the experiments and analysis.

Funding

This work is supported by the National Natural Science Foundation of China (No. 42174050, 62172066, 62172064, 62322601), National Science Foundation for Excellent Young Scholars (No. 62322601), Open Research Projects of Zhejiang Lab (No. K2022NB0AB07), Venture & Innovation Support Program for Chongqing Overseas Returnees (No. cx2021047), Chongqing Startup Project for Doctorate Scholars (No. CSTB2022BSXM-JSX005), Excellent Youth Foundation of Chongqing (No. CSTB2023NSCQJQX0025) , China Postdoctoral Science Foundation (No. 2023M740402), and Fundamental Research Funds for the Central Universities (No. 2023CDJXY-038, 2023CDJXY-039).

Availability of data and materials

The datasets generated and/or analysed during the current study are available online via https://archive.ics.uci.edu/dataset/310/ujiindoorloc.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 7 July 2023 Accepted: 19 December 2023
Published online: 11 March 2024

References

Bhargava, P., Krishnamoorthy, S., Shrivastava, A., Nakshathri, A. K., Mah, M., & Agrawala, A. (2015). Locus: Robust and calibration-free indoor localization, tracking and navigation for multi-story buildings. *Journal of Location Based services, 9*(3), 187–208.

Bianchi, F. M., Grattarola, D., Livi, L., & Alippi, C. (2021). Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*, 3496.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. In *International conference on learning representations (ICLR 2014)*.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems* (pp. 3844–3852).

Deldjoo, Y., Schedl, M., Cremonesi, P., & Pasi, G. (2020). Recommender systems leveraging multimedia content. *ACM Computing Surveys (CSUR), 53*(5), 1–38.

Du, J., Zhang, S., Wu, G., Moura, J. M., & Kar, S. (2018). Topology adaptive graph convolutional networks. In *International conference on learning representations (ICLR 2018)*.

Elbakly, R., Aly, H., & Youssef, M. (2018). Truestory: Accurate and robust rf-based floor estimation for challenging indoor environments. *IEEE Sensors Journal, 18*(24), 10115–10124.

Elbakly, R., & Youssef, M. (2020). The storyteller: Scalable building-and ap-independent deep learning-based floor prediction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 4*(1), 1–20.

El-Sheimy, N., & Li, Y. (2021). Indoor navigation: State of the art and future trends. *Satellite Navigation, 2*(1), 1–23.

Gerstner, W. (1995). Time structure of the activity in neural network models. *Physical Review E, 51*, 738–758. https://doi.org/10.1103/PhysRevE.51.738

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323).

Gu, F., Blankenbach, J., Khoshelham, K., Grottke, J., & Valaee, S. (2019). Zeefi: zero-effort floor identification with deep learning for indoor localization. In *IEEE global communications conference (GlobeCom)*

Gu, F., Hu, X., Ramezani, M., Acharya, D., Khoshelham, K., Valaee, S., & Shang, J. (2019). Indoor localization improved by spatial context-a survey. *ACM Computing Surveys, 52*(3), 64:1-64:35. https://doi.org/10.1145/3322241

Gu, F., Khoshelham, K., Valaee, S., Shang, J., & Zhang, R. (2018). Locomotion activity recognition using stacked denoising autoencoders. *IEEE Internet of Things Journal, 5*(3), 2085–2093.

Gu, F., Khoshelham, K., Yu, C., & Shang, J. (2018). Accurate step length estimation for pedestrian dead reckoning localization using stacked autoencoders. *IEEE Transactions on Instrumentation and Measurement, 68*, 2705.

Gu, F., Sng, W., Taunyazov, T., & Soh, H. (2020). Tactilesgnet: A spiking graph neural network for event-based tactile object recognition. In *2020 IEEE/RSJ International conference on intelligent robots and systems (IROS)* (pp. 9876–9882).

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 639–648).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *J. international conference on learning representations (ICLR 2017)*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60* (6), 84–90.

Li, Y., Gu, C., Dullien, T., Vinyals, O., & Kohli, P. (2019). Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning* (pp. 3835–3845).

Luo, J., Zhang, Z., Wang, C., Liu, C., & Xiao, D. (2019). Indoor multifloor localization method based on wifi fingerprints and lda. *IEEE Transactions on Industrial Informatics, 15*(9), 5225–5234. https://doi.org/10.1109/TII.2019.2912055

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the thirtieth international conference on machine learning (ICML)* (Vol. 30).

Qi, H., Wang, Y., Bi, J., Cao, H., & Si, M. (2019). Fast floor identification method based on confidence interval of wi-fi signals. *Acta Geodaetica et Geophysica, 54*(3), 425–443.

Roy, K., Jaiswal, A., & Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature, 575*(7784), 607–617.

Shen, X., Chen, Y., Zhang, J., Wang, L., Dai, G., & He, T. (2015). Barfi: Barometer-aided wi-fi floor localization using crowdsourcing. In *2015 IEEE 12th international conference on mobile Ad Hoc and sensor systems (MASS),* (pp. 416–424).

Shrestha, S. B., & Orchard, G. (2018). Slayer: Spike layer error reassignment in time. In *Advances in neural information processing systems* (pp. 1412–1421).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations (ICLR 2015)*.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., & Bloom-Ackermann, Z. (2020). A deep learning approach to antibiotic discovery. *Cell, 180*(4), 688–702.

Torres-Sospedra, J., Montoliu, R., Martínez-Usó, A., Avariento, J. P., Arnau, T. J., Benedito-Bordonau, M., & Huerta, J. (2014). 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)* (pp. 261–270).

Torres-Sospedra, J., Montoliu, R., Trilles, S., Belmonte, Ó., & Huerta, J. (2015). Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems. *Expert Systems with Applications, 42*(23), 9263–9278.

Varshavsky, A., LaMarca, A., Hightower, J., & De Lara, E. (2007). The skyloc floor localization system. In *Fifth annual IEEE international conference on pervasive computing and communications, 2007. PerCom'07* (pp. 125–134).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Wang, X., Gao, L., Mao, S., & Pandey, S. (2016). Csi-based fingerprinting for indoor localization: A deep learning approach. *IEEE Transactions on Vehicular Technology, 66*(1), 763–776.

Weinlich, M., Kurz, P., Blau, M. B., Walcher, F., & Piatek, S. (2018). Significant acceleration of emergency response using smartphone geolocation data and a worldwide emergency call support system. *PloS One, 13*(5), e0196336.

Wu, Y., Deng, L., Li, G., Zhu, J., & Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience, 12*(331), 1–12.

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., & Shi, L. (2019). Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 1311–1318).

Ye, H., Gu, T., Tao, X., & Lu, J. (2014). F-loc: Floor localization via crowdsourcing. In *20th IEEE International Conference on parallel and distributed systems (ICPADS)* (2014) (pp. 47–54).

Ye, H., Gu, T., Tao, X., & Lu, J. (2016). Scalable floor localization using barometer on smartphone. *Wireless Communications and Mobile Computing, 16*(16), 2557–2571.

Ye, H., Gu, T., Zhu, X., Xu, J., Tao, X., Lu, J., & Jin, N. (2012). Ftrack: Infrastructure-free floor localization via mobile phone sensing. In *IEEE international conference on pervasive computing and communications (PerCom)*, 2012 (pp. 2–10).

Zhang, Y., Ma, L., Wang, B., & Qin, D. (2020). Building floor identification method based on dae-lstm in cellular network. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)* (pp. 1–5).

Zhao, H., Shang, J., Liu, K., Chen, C., & Gu, F. (2023). Edgevo: An efficient and accurate edge-based visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Zhao, Y., Gong, W., Li, L., Zhang, B., & Li, C. (2023). An efficient and robust fingerprint based localization method for multi floor indoor environment. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2023.3298603

Zheng, H., Wu, Y., Deng, L., Hu, Y., & Li, G. (2021). Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, pp. 11062–11070).

Zhou, B., Gu, Z., Gu, F., Wu, P., Yang, C., Liu, X., Li, L., Li, Y., & Li, Q. (2022). Deepvip: Deep learning-based vehicle indoor positioning using smartphones. *IEEE Transactions on Vehicular Technology, 71*(12), 13299–13309.

Zhuang, Y., Hua, L., Qi, L., Yang, J., Cao, P., Cao, Y., Wu, Y., Thompson, J., & Haas, H. (2018). A survey of positioning systems using visible led lights. *IEEE Communications Surveys & Tutorials, 20*(3), 1963–1988.

Publisher's Note