ISSN 2723-6471

Applying Transfer Learning on Various GNN Model Training in Indoor Positioning System Tasks

Kevin Wijaya[1,*], Hanif Muhammad Sangga Buana[1], Gede Putra Kusuma[1]

[1]*Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480*

(Received: July 4, 2024; Revised: August 31, 2024; Accepted: September 11, 2024; Available online: September 23, 2024)

Abstract

Determining location and orientation has always been a fundamental challenge, driving advances from maps and compasses to modern global navigation satellite systems (GNSS). However, GNSS performs poorly indoors due to signal attenuation and lack of elevation accuracy, necessitating the development of indoor positioning systems (IPS). Various technologies such as Wi-Fi, Bluetooth Low Energy (BLE), and RFID have been deployed, typically relying on received signal strength (RSS) and fingerprinting to improve accuracy. While previous research focused on training a single model for an entire building, this study explores the creation of floor-specific models by applying transfer learning to various GNN models. This is done to address the substantial signal distortion between floors. Using the UTSIndoorLoc dataset, we evaluate Graph Attention Network (GAT), GraphSAGE, and Graph Convolutional Network (GraphConv) for predicting two-dimensional indoor positions based on RSSI fingerprints. We propose 2 transfer learning model training methods, Schema A and Schema B. Schema A trains the base model iteratively through each floor, and Schema B trains the base model on a unified dataset. Schema B with GraphConv achieved the best results with a mean positioning error of 6.2176 meters. Whilst Schema A achieved a best-case mean positioning error of 6.3900 meters. Both outperforming the standard unified model which has a mean positioning error of 8.0808 meters.

Keywords: Graph Neural Network, Transfer Learning, KNN Graph, Indoor Positioning System, Graph Attention Network, Graph Convolutional Network, GraphSAGE, Wi-Fi Fingerprinting

1. Introduction

Determining location and direction is a fundamental requirement for navigation, spurring the development of various technologies from maps, compasses, lighthouses, radars, and beyond [1]. In the 21st century, we typically rely on Global Navigation Satellite Systems (GNSS) such as Global Positioning System (GPS), which are accessed by various electronic devices to determine a location in relation to the world. However, GNSS is not suitable for use in indoor environments due to a phenomenon known as signal attenuation which is a form of signal interference caused by having to pass through various building materials [2], and its poor accuracy in determining altitude [3] that is crucial for applications in multi-story buildings. This weakness leads people to create various Indoor Positioning Systems (IPS) to track objects in indoor environments. Indoor tracking has been used in many applications and environments to improve its operation such as malls [4], airports [5], universities [6], [7], elderly housing [8], etc.

In previous research, many technologies can and have been used as the foundation of an IPS, such as Wi-Fi [6], Bluetooth Low Energy (BLE) [8], Radio Frequency Identification (RFID) [9], or some combinations thereof [7]. They mainly work by transmitting and receiving signals and then interpreting the characteristics of the signals to determine the receiver's position in relation to the beacon. One of the most common and simple ways to do this is by using Received Signal Strength (RSS). RSS measures the signal strength of various transmitters received by a receiver (such as a cellphone). The stronger the signal, the closer the distance between a transmitter and a receiver [10]. Using this data and sufficient enough transmitters spread around an area we could locate the position of an object using trilateration. However, RSS has several problems, such as signal irregularity due to walls blocking signals between the receiver and transmitter. A phenomenon known as the multipath effect also makes signals received by the receiver

***Corresponding author: Kevin Wijaya (kevin.wijaya018@binus.ac.id)**
DOI: https://doi.org/10.47738/jads.v5i2.XXX

 ISSN 2723-6471

inconsistent. To mitigate this problem a method known as fingerprinting has been introduced [11]. Fingerprinting works by creating a radio map of signals received in various parts of a building to simulate real-time conditions and training a machine-learning model on it in order to achieve better positioning accuracy [12].

Most research up to this point has focused on building one model to predict positions [6], [7], [8], [9] for an entire building. While this approach is much simpler, it implicitly assumes that signals received on sensors from different floors have the same importance in localization accuracy. However, empirical measurements in indoor environments consistently shown that wireless signals experience significant attenuation (distortion) when passing through solid objects such as floors, resulting in much weaker and noisy RSS values compared to signals received by sensors on the same floor [13]. This means that a large portion of the input features bear little importance and may even be detrimental to accurately determining an object's position within a given floor.

This mismatch can negatively affect model performance. A unified model risks becoming too generalized as it must simultaneously balance the importance of each sensor per floor with capturing each floor's signal-location relationships that differ from each other. This can lead the model to over-regularize and achieve suboptimal feature utilization. Training floor-specific models offers a more focused way for the model to learn floor-specific signal-location relationships. To retain useful cross-floor information this paper will explore the benefits or lack thereof of using a transfer learning method to share learned insights between floors.

This paper proposes the use of Graph Neural Networks (GNN) to achieve IPS as this method has demonstrated merit in previous IPS research [9] and applications in other sectors. We aim to further enhance this by using a Graph Attention Network (GAT) to use the attention mechanism it provides to focus on certain sensors that matter.

Section 2 contains a summary of other relevant studies on IPS. Section 3 details methods and resources that are used in this study such as dataset, GNN, Fingerprinting, Transfer Learning, etc. Meanwhile, Section 4 discusses and lays out the results of the study and what we thought of it in general. Section 5 serves to summarize this study and give directions for improvements that could be made in future works.

2. Literature Review

IPS has become more important in daily life or work applications such as navigation, resource tracking, and smart environments. Traditional IPS has some approaches that rely on Wi-Fi, Bluetooth, and hybrid methods that will be used to get data fingerprinting [7], [11]. These methods often experience environmental challenges such as signal attenuation and dynamic obstacles, and because of that this method only obtains moderate accuracy. To overcome this, researchers came up with the use of graph-based machine learning methods. The result of this machine learning method including GNN has shown promise in overcoming these limitations, because of their ability to model complex relational data effectively [12], [13].

2.1. Graph Neural Networks

GNNs has become popular in indoor localization, because of their ability to encode spatial and temporal relationships in graph-structured data. Research that has efficiently localized floors in multi-story buildings highlights the scalability of GNN for large indoor spaces using spike GNN [13]. Another research has captured higher-dimensional relationships for increased positioning accuracy using high-order GNN [14]. Similar to previous research that effectively models indoor environments using Wi-Fi RSSI data and makes substantial improvements in localization accuracy [12].

Different GNN architectures offers different strengths and weaknesses that are interesting to explore in indoor positioning tasks. Graph Convolutional Network (GraphConv) performs neighborhood aggregation, which can help stabilize learning and reduces sensitivity to noisy RSSI measurements, but this can also lead to an oversmoothed RSSI features rendering it unable to capture finer details [17]. GraphSAGE improves scalability and adds inductive learning through neighborhood sampling, making it great for large environments, however its fixed aggregation functions can make it take in weak or noisy RSSI measurements [18]. While Graph Attention Network (GAT) introduces attention mechanisms that weigh nodes based on their relevance, this can lead to both a more focused model and a persistent overfitting problem [19].

2.2. Transfer Learning

In IPS, signal distributions often change due to sensor placements, environmental factors, or physical barriers such as a building floor. This results in de-facto different domains that may change in a given dataset. In this case, RSSI fingerprints collected on different floors are likely to follow statistically different distributions, even with the same set of wireless access points.

Transfer learning has emerged as a valuable technique to overcome these domain adaptation challenges. This is done by transferring knowledge from a model trained on a more general source dataset to more specialized model to give more context and thus improve performance compared to just training on the specialized dataset. A general research on transfer learning in GNN showed that transfer learning is effective in improving knowledge transfer and graph representation, thus increasing performance [15]. Based on this study, we can enhance the robustness of GNN-based IPS using the potential of transfer learning.

2.3. Challenges and Research Gaps

Developing floor-specific models for indoor localization systems using GNN and transfer learning is challenging due to the different signal characteristics between floors in multi-story buildings due to disturbances such as floor attenuation and multipath interference. While GNN has been widely used in indoor positioning systems, most studies train a single unified model for an entire building which trains the model on a lot of noise for a given floor. This approach risks learning needlessly overgeneralized representations that do not adequately capture the fine-grained, floor-specific RSSI variations.

Existing methods in IPS like domain-invariant GNN [21] tries to address the general domain adaptation problem by encouraging the graph representations to be statistically similar across different domains through adversarial training objectives. However, this can lead to less attention being paid to preserving domain-specific characteristics that are informative for position localization. Scalable GNN frameworks [13] and multimodal methods [17] show potential but have not yet fully integrated floor-specific transfer learning and/or training. Advancing this area requires robust methods that balance accuracy and efficiency, supported by standardized multi-story datasets such as UTSIndoorLoc [6].

In table 1 the application of graph-based methodologies, particularly GNN, has shown promise in indoor localization due to their ability to model complex spatial and relational data. Studies such as [14] and [16] have highlighted the efficacy of GNN in learning spatial representations for indoor positioning systems (IPS). Furthermore, enhancements like Graph Attention Networks (GATs), as proposed by [19], have been instrumental in improving performance by focusing on relevant spatial features, echoing the attention-based approaches outlined in [22]. Transfer learning, as elaborated by [26] and [20], provides a framework for leveraging pre-trained models to address data scarcity across domains, aligning well with RSSI-based IPS tasks. Additionally, the transformation of RSSI data into graph structures, as discussed by [8] and [27], ensures an accurate representation of spatial relationships. These insights collectively justify the exploration of GNN with transfer learning for enhancing indoor localization accuracy.

3. Methodology

3.1. Project Workflow

The workflow diagram in figure 1 represents our approach to completing this project. In broad strokes, we first process our data into graphs, train our general model on the training graph data, and then use that general model as a base to train floor-specific models. After the training procedure is done the resulting model will then be matched against the test graph data and evaluated based on its average error.

 ISSN 2723-6471

Table 1. Previous Works Summary

Title	Method/Model	Result (Accuracy/Positioning Error)	Dataset	Notes
Adaptive Transfer Learning on Graph Neural Networks [15]	Adaptive Auxiliary Loss Weighting (AUX-TS) with Graph Neural Networks (GNNs)	Improved accuracy across dataset for node classification and link prediction	• Node Classification: OAG_CS, Reddit • Link Prediction: Last-FM, Book-Crossing	Adaptively selects and combines auxiliary tasks using gradient similarity and meta-learning to improve transfer learning effectiveness.
Inductive Representation Learning on Large Graphs [18]	GraphSAGE (GNN)	Not explicitly stated / N/A	Not specified	Introduced GraphSAGE for inductive representation learning.
IndoorGNN: A Graph Neural Network based approach for Indoor Localization using WiFi RSSI [19]	Graph Neural Network	UjiIndoorLoc: 95.8% accuracy; MNAV: 97.5% accuracy	UjiIndoorLoc and MNAV	Wifi RSSI data is divided into regions (for example each floor has 6 regions) and the task is to classify them based on regions
Accurate and Efficient Floor Localization with Scalable Spiking Graph Neural Networks [13]	Spiking Graph Neural Networks (FloorLocator)	Average 89% Accuracy	UJIIndoorLoc	Proposed a scalable approach for floor-level localization.
Indoor Localization Algorithm Based on a High-Order Graph Neural Network [14]	High-order GNN	SoLoc: 5.81 m; Self-Built: 1.29 m	SoLoc and the Self-Built Underground Garage Dataset	Utilized high-order graph structures for improved accuracy.
Indoor Localization using Graph Neural Networks [12]	GNNs	UjiIndoorLoc: 92%; MNAV: 97.2%	UJIIndoorLoc and Museo Nacional de Artes Visuales (MNAV, National Museum of Visual Arts)	Focused on applying GNNs to indoor localization problems.
A GNN-based Indoor Localization Method Using Mobile RFID Platform [9]	GNN	GCN: 5.87 cm; GraphSAGE: 5.74 cm; GIN: 6.02 cm	RFID data	Applied GNNs with RFID platforms for indoor localization.
A Novel Convolutional Neural Network Based Indoor Localization Framework with WiFi Fingerprinting [6]	Convolutional Neural Network (CNN)	95.46% floor hit rate, 7.6 m positioning error (MAE)	UTSIndoorLoc	Proposed CNNLoc model; Made new dataset (UTSIndoorLoc).
A Multimodal Graph Fingerprinting Method for Indoor Positioning Systems [22]	Multimodal Graph Neural Network	All Feature: 3.22; WIFI + EMF: 2.87	Custom Dataset (in university room)	Combined multimodal data with GNNs for enhanced fingerprinting.
Domain Adversarial Graph Convolutional Network Based on RSSI and Crowdsensing [21]	WiAGCN and WiDAGCN	WiAGCN: 5.32 m; WiDAGCN: 4.84 m	Microsoft Indoor Location Competition 2.0	Used RSSI and crowdsourced data for domain adaptation in indoor localization.
A Geometric Deep Learning Framework for Accurate Indoor Localization [24]	Geometric Deep Learning Framework	Custom Dataset: Single: 1.1145; Combination: 0.9309 SoLoc dataset: Single: 3.5156; Combination: 2.9642	SoLoc dataset and Custom from other paper	Utilized geometric principles for high localization accuracy.
Graph-Based Machine Learning for Practical Indoor Localization [25]	Graph-based indoor localization (GBIL) [system to apply GCN and VDL to CSI]	1.19 m	Custom Dataset (15x20 m Room)	Focused on practical considerations for implementing graph-based localization methods.
Indoor positioning system using hybrid method of fingerprinting and pedestrian dead reckoning [7]	Kalman Filter, ANN, and SVR	111.78 cm mean error (ANN)	Custom Dataset (in university room)	Combining basic RSSI fingerprint with pedestrian dead reckoning (data from phone sensors such as gyroscope) to better track object position.
A Low Cost Indoor Positioning System Using Bluetooth Low Energy [8]	Multiple, but Random Forest is best	99.74% accuracy (predicting location that is divided into grids)	Custom Dataset (in old people's house)	Making RSSI based BLE IPS prediction in old people's home.
Smartphone-Based Indoor Localization with Bluetooth Low Energy Beacons [11]	Polynomial regression model (PRM), channel-separate fingerprinting (FP), extended Kalman filtering (EKF), and outlier detection	Accuracies of <3.1 m at 90% of the time	Custom Dataset (Office Environment)	Experimenting with BLE density and their proposed algorithm.

 ISSN 2723-6471

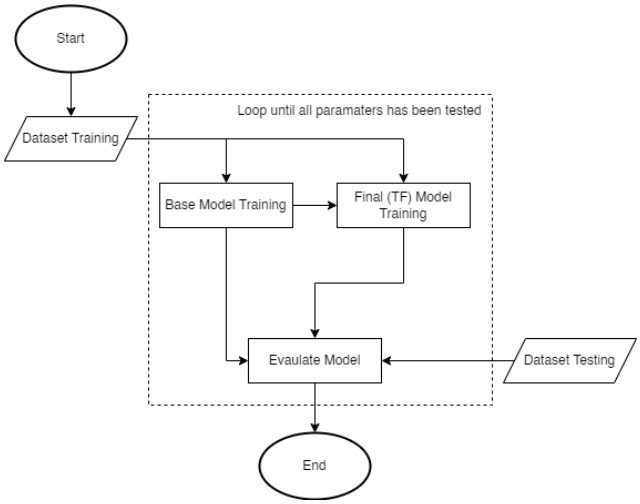

Figure 1. Project Workflow

3.2.Dataset and Pre-Processing

The dataset input that we use for this study comes from [6] called UTSIndoorLoc. It is an RSSI radio map dataset taken in the FEIT Building at the University of Technology Sydney (UTS). As can be seen from table 2, this dataset is composed of 589 Wireless Access Points (WAPs) divided into 16 floors containing 9496 sample data. It is split into 9108 samples for training and 388 samples for testing. The value of Wi-Fi RSSI for this dataset goes from 0 (strongest) to -96 (weakest) dBm. WAP that does not detect anything is listed with the value 100.

Table 2. Example of UTSIndoorLoc Dataset

WAP1	WAP2	WAP(n)	WAP589	Pos_x	Pos_y	Floor_ID
100	-84	…	100	390.071	181.088	8
-80	-81	…	100	328.784	179.397	8
-88	100	…	100	389.625	732.924	8

This dataset has some quirks such as containing a significant number of duplicate data. Without these duplicates, the amount of data usable decreases to 2341 (1955 training & 386 testing). We also would like to examine replacing the value 100 (not detected) with a number 1 dBm below the lowest RSSI value and applying normalization to it as has been suggested by [28]. Table 3 presents our analysis of the dataset regarding the amount of relevant sensors per floor. We chose -85 dB as our threshold because other research has demonstrated that Wi-Fi RSSI values below -85 are unreliable to be used as guides for positioning [29]. This is our best approximation of the amount of sensors per floor as there is no metadata about which sensors are placed within what floor in the original dataset or research for UTSIndoorLoc. We found that in general each floor can only benefit from between 2.86% to 5.96% sensors with an average of 4.30%. These percentages indicate that not even all the sensors on a given floor can help in the localization of a position, let alone sensors from a different floor. This preliminary analysis strengthens the case for building a floor-specific model.

Table 3. Relevant RSSI Percentages (>-85dB)

Floors	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12
RSSI >-85 dB (%)	4.33	2.98	3.45	2.86	3.18	4.44	3.45	4.54	4.16	4.53	3.25	3.22	4.79	4.86	5.45	5.96
Mean (%)	4.30															

In this study, we are purely concerned with predicting the x and y positions. We thus will disregard any data that is not relevant such as time, building, etc. that exists on the original dataset. The model will process WAP and Floor_ID as

 ISSN 2723-6471

input variables and output x and y positions. Before the input variables are fed into the model, they must first be turned into a graph for reasons that will be discussed in Section 3.3.

3.3. Graph Neural Network

Graph neural network is a class of deep learning models that processes structured data using graphs. GNN works by aggregating and updating node features through iterative message-passing mechanisms, enabling them to capture complex relationships between nodes and edges in graph data [30]. We have chosen to explore GNN further in this study because of its good performance in previous IPS research [23].

Figure 2 represents the model architecture that we are using in this experiment. We employ a basic standard 5-layer convolutional component followed by a 4-layer MLP, maintaining this fixed structure across all experiments. The first layer is a convolutional layer that takes 590 input nodes, representing the combination of WAP and Floor_ID features, through smaller hidden channels to capture multi-hop spatial relationships. Each node signifies an RSSI fingerprint, whereas the edges represent similarity-based neighborhood relationships among spatially related locations. The quantity of nodes is determined by the data, specifically the density of fingerprints per floor, thereby ensuring spatially consistent message transmission. The default hidden channel size is established at 128 as we felt it offers a good balance between representational capacity and generalization, considering the high dimensionality of the dataset. This is followed by an activation layer utilizing the LeakyReLU() function to introduce non-linearity, which helps prevent dead neurons and improve training stability [31]. Next, a batch normalization layer is applied to normalize the data from the input layer, ensuring more stable learning and reducing sensitivity to outliers. To mitigate overfitting, a dropout layer is included, which randomly ignores some neurons during training, encouraging the model to generalize better. Finally, the second convolutional layer produces hidden channel outputs that are connected to a Multi-Layer Perceptron (MLP) for further feature extraction.

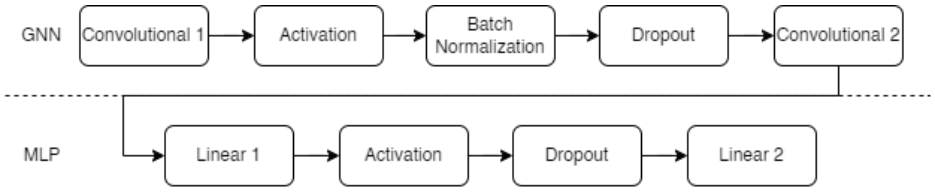

Figure 2. GNN Architecture

The 4-layer MLP component is used to map spatial features to 2D coordinates through the hidden units. The first layer is a linear transformation layer that receives input from the second convolutional layer and outputs data with the same channel dimension. This is followed by an activation layer using the LeakyReLU() function, with a default negative slope of 0.2 to maintain non-linearity and prevent neuron inactivity [31]. Another dropout layer is applied afterward with a dropout probability of 0.5 to further reduce overfitting by forcing the network to rely less on specific neurons. Finally, the second linear layer generates the model's final output, completing the MLP structure.

The architecture depth and dimensions are optimized for spatial modeling, convolutional layers capture local signal dependencies while the MLP performs spatial-to-coordinate regression. Hidden dimensions follow a structure that consistently gets smaller to progressively refine spatial representations. This design aims to create a simple, relatively lightweight model, that can still provide good capability and generalization for indoor positioning tasks.

3.3.1. K-Nearest Neigbor (KNN) Graph

Graph Neural Network as in the name requires graph as data input instead of regular ASCII-based data. This necessitates the transformation of ASCII-based data to some kind of graph. The graph that we have chosen in this is a spatial KNN graph. This choice is motivated by the fact that spatial KNN graph effectively represents the connection between different points in space that the data express [21]. KNN graph is a graph structure that is used to present data points based on distance or similarity with a certain amount (k) connection considered per node [27]. It also has been used extensively to great results in many different studies [14], [15], [16], [21].

 ISSN 2723-6471

Figure 3 represents the general architecture of the graph that we are using. Each data point will be connected to several of its closest edge nodes depending on the k value. The edge node is separated from the other data and will remain the same consistent across the training, validation, and testing dataset. In the case of the GAT and GraphSAGE models, the graph is treated as unweighted, whereas the GraphConv model incorporates edge weights derived from distance or similarity measures, as it inherently supports weighted graphs. Various values of k will be assessed to identify the optimal neighborhood size.

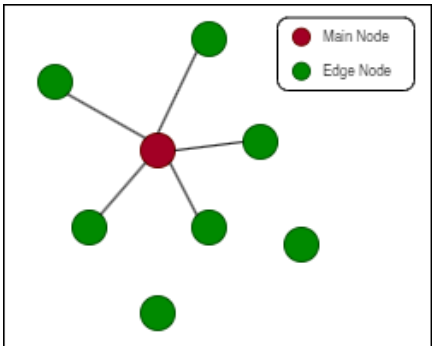

Figure 3. KNN Graph (k=5) Example

In addition to Euclidean distance, cosine similarity is examined as an alternative metric, as it has been demonstrated to perform more effectively in high-dimensional feature spaces [32]. By prioritizing angular similarity over absolute magnitude, cosine similarity can offer enhanced robustness against scale variations and alleviates the curse of dimensionality, resulting in more stable and meaningful neighborhood relationships.

3.3.2. Graph Attention Network (GAT)

GAT is a modification of traditional GNN by adding a learnable attention mechanism. This allows the model to automatically assign different importance weights on neighboring nodes during aggregation, allowing it to focus on relevant parts of the input [19]. We hypothesized that this mechanism would perform well in dealing with the limited amount of useful data in RSSI-based fingerprinting IPS as it allows the model to focus its attention on the sensors that are relevant to the detection of an object on a particular floor.

Attention weights in GAT are initialized using Xavier/Glorot uniform initialization for both the linear transformation matrices and attention parameters. Specifically, each attention head's weight matrix and attention vector are initialized from a uniform distribution to ensure stable gradient flow during early training. Additionally, dropout is applied to the attention coefficients and feature transformations as a regularization technique to avoid overfitting. The attention heads and hyperparameters will be fine-tuned to enhance localization accuracy.

3.3.3. Graph Sample and Aggregate (GraphSAGE)

GraphSAGE is an extension of traditional GNN designed to handle large-scale graphs by using a sampling-based approach instead of full neighborhood aggregation [18]. It randomly samples a fixed number of neighbors for each node. This makes it more computationally efficient, as it avoids excessive memory usage and computational cost in dense graphs.

We hypothesize that GraphSAGE's sampling mechanism could be beneficial for RSSI-based fingerprinting IPS, as it prevents overwhelming the model with noise from distant, less relevant sensors. By sampling and aggregating only the most relevant sensor signals, GraphSAGE can help focus on local spatial dependencies, improving model efficiency and generalization. We will try to fine tune this aggregation mechanism to deliver the best outcome. In this study, GraphSAGE is implemented only using the default uniform sampling strategy from the original paper, where neighbors are randomly selected with equal probability. This approach introduces beneficial stochasticity during training, thereby theoretically improving generalization across varying sensor configurations. While alternative strategies like importance-based sampling could prioritize high-RSSI sensors, we have decided to not widen the scope of the study any further due to time and computational constraints.

 ISSN 2723-6471

3.3.4. Graph Convolutional Network (GCN/GraphConv)

GCN [17] extend traditional convolutional neural networks (CNNs) to operate on graph-structured data. GCN uses spectral graph convolutions, which propagate node features by performing weighted aggregations from their neighbors.

In our case, GCN could be useful for RSSI-based fingerprinting IPS as it enforces a smoothing effect, meaning that node features become more similar to their neighbors over layers. This can be useful when modeling sensor data, as signals from nearby access points often have similar patterns, helping the model learn spatial dependencies more effectively. GCN are also able to take in edge weights which we will assign based on the similarity metric chosen to maximize information given to the model. Specifically, we intend to compare and implement two weighting schemes: (1) raw cosine similarity values ranging [0, 1] where higher weights indicate stronger signal correlation, and (2) inverse Euclidean distance where weights represent spatial proximity between sensor readings. These un-normalized raw weights are then passed directly to the GCN layers. This approach provides interpretable edge weights that directly correspond to measurable signal characteristics, enhancing model transparency while capturing meaningful sensor relationships. We will also try to fine tune the aggregation mechanism of GCN to deliver the best outcome.

3.4. Transfer Learning (Model Training)

Transfer learning is a method to transfer knowledge between similar tasks [26]. In machine learning, this usually manifests as taking a model trained on a similar dataset for similar tasks and retraining it to fit with another dataset or task. The benefit of this approach is that the model needs fewer specific datasets to be trained in to achieve a satisfactory result. In the case of GNN, it also reduces the graph size, thereby reducing the total memory needed to process it, which can be a limitation depending on hardware, data size, and graph type. For this study, we proposed two base models for the transfer learning training schema.

Schema A as illustrated on figure 4, splits the dataset based on floors and then trains the model on the first floor before transferring to the next and so on until all the data has already been part of the base model training. Meanwhile, Schema B as illustrated on figure 5, trains the base model conventionally on data from all floors at once.

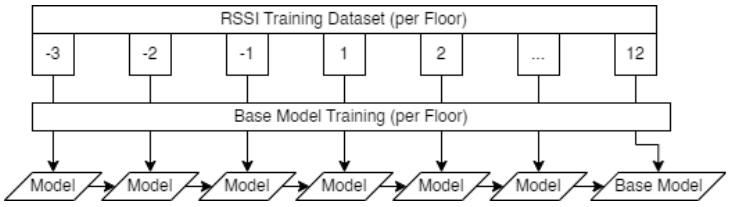

Figure 4. Transfer Learning Base Model A Training Schema

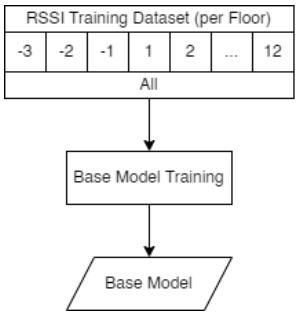

Figure 5. Transfer Learning Base Model B Training Schema

Following the training of the base model, a distinct model is fine-tuned for each floor (see figure 6). Each floor-specific model is initialized using the weights from the base model and is trained end-to-end with all layers unfrozen. A learning rate scheduler is employed to halve the learning rate every 100 epochs in order to stabilize convergence, and no parameter resets are implemented between the different floors. All model training is conducted up to 500 epochs. Various optimizers and learning rates are assessed to determine the optimal configuration.

 ISSN 2723-6471

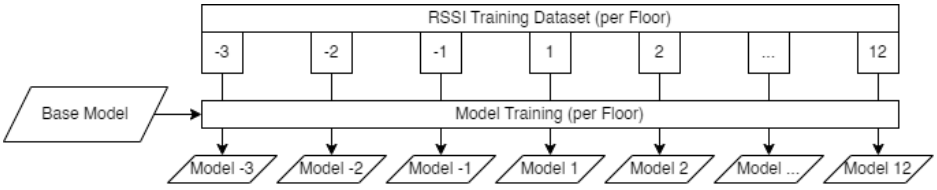

Figure 6. Final Transfer Learning Model

Even though we initially concocted base model A first, after much discussion we hypothesized that re-training from base model B might yield a superior result compared to base model A. This is because base model B might provide a more neutral base as it is trained with all the floors at once, meanwhile base model A might overly emphasize and remember the data from the latter floors compared to earlier floors.

3.5. Equations

We are principally looking to determine the positioning error of the predicted output from the model compared to the actual values from the dataset. We used the mean positioning error as the evaluation metric as this is the main evaluation metric used in other IPS research. The positioning error itself will be calculated using Euclidean distance based on the Pythagoras theorem as illustrated on (1). All the positioning errors will then be averaged to gain the mean positioning error as illustrated on (2).

$$e_i = \sqrt{(xpred_i - xtrue_i)^2 + (ypred_i - ytrue_i)^2} \quad (1)$$

$$E_{mean} = Mean(e_1, e_2, \dots, e_n) \quad (2)$$

4. Results and Discussion

The initial baseline configuration and parameters at the start of this experiment are GAT model using Schema B transfer learning with 3 attention head, 64 hidden channels, 0.2 negative slope, 0.01 learning rate with Adam optimizer, and 0.3 dropout rate. The process of model training adheres to a predetermined epoch schedule, employing a step-based learning rate scheduler that reduces the learning rate by half every 100 epochs to enhance convergence. Generalization is assessed through validation loss, while dropout and batch normalization serve as regularization techniques to alleviate overfitting. Instead of using early stopping functions, we chose to fully train the model through all 500 epochs and choose the epoch with least loss to test. After every successive result we will fine-tune the parameters according to the best parameters found in the previous experiments. All results reported with a standard deviation were run five times and the numbers correspond to mean ± standard deviation (σ). While all other results were run three times, and the best mean positioning error (MPE) was selected.

4.1. Data Pre-Processing Tuning

As can be seen from table 4, we found that the removal of duplicate data negatively impacts the model quality. This is most likely because duplicate data creates a more balanced training dataset, as the original values on the dataset do not form a balanced distribution. This should not be unfair on the test results as the vast majority of duplicates exist on the training and validation dataset, not on the testing dataset. This led us to refrain from doing it in our subsequent experiments.

Table 4. Duplicate Data Removal Result

	Original	Duplicate Removed
Positioning Error	6.9301 ± 0.12	7.1400 ± 0.10

Table 5. Data Pre-processing Fine-tuning Result

	Without Pre-Processing	Changing Undetected RSSI Value	Changing Undetected RSSI Value and Normalization
Positioning Error	6.9301 ± 0.12	7.3288 ± 0.08	7.0998 ± 0.09

 ISSN 2723-6471

As can be seen in table 4, any kind of data pre-processing harms the resulting model accuracy. We found that the removal of duplicate data negatively impacts the model quality, so we refrained from doing it in our next experiments. After conducting further testing, we found that doing any kind of data pre-processing harms the resulting model accuracy. As can be seen on table 5, the observed differences here are larger than the corresponding standard deviations and remain consistent across multiple runs, indicating a degradation in performance due to preprocessing. This result contradicts the findings of another research [28] that concludes replacing undetected RSSI value to -97 and applying normalization increases model performance. These findings led to us not performing any data pre-processing to the original dataset.

4.2. KNN Graph Tuning

From table 6, we experimented with different values of k in the K-nearest neighbor (KNN) graph to determine the optimal number of neighbors for constructing the graph. The positioning error fluctuates slightly between different values with 5 being the k value with the lowest positioning error (6.7753). This finding suggests that a k value of 5 provided a good balance between having enough neighbors to gain enough information while avoiding excessive noise from irrelevant data points.

Table 6. K-nearest Neighbor Fine-tuning Results

Nearest Neighbor	3	4	5	6	7	8	9	10
Positioning Error	6.7902	6.8574	6.7753	6.7862	6.8093	6.8768	6.8093	6.7808

Table 7 compares the result of two different distance metric: Euclidean Distance and Cosine Similarity. We found that Cosine Similarity performed better than Euclidean Distance (6.7753 vs 6.8582), possibly indicating that RSSI-based positioning benefits from similarity-based metrics rather than pure distance calculations.

Table 7. Distance Metric Fine-tuning Results

	Euclidean Distance	**Cosine Similarity**
Positioning Error	6.8582	6.7753

4.3. Model Attention and Aggregation Tuning

In table 8, we attempt to fine-tune the number of attention heads for Graph Attention Network (GAT). The results show that increasing the number of attention heads improve performance up to 4 attention heads (6.609). However, beyond this we notice a degradation in performance, possible due to excessive parameters causing overfitting. This could indicate that a mixture of 3 sensors for triangulation and floor data is optimal for detecting a position without overcomplicating the model.

As for GraphSAGE and GraphConv (table 9), we tested several different aggregation strategies: Add, Mean, and Max. Both models perform best on mean aggregation, with the lowest positioning error for GraphSAGE at 6.7011 and GraphConv at 6.6571. This suggests that averaging the feature information from neighboring nodes provides more stable and informative representations compared to simply summing (Add) or selecting the maximum value (Max).

Table 8. GAT Attention Head Fine-tuning Results

Attention Head	1	2	3	4	5
Positioning Error	6.8556	6.8221	6.7753	6.6609	6.8202

Table 9. GraphSAGE and GraphConv Aggregation Method Fine-tuning Results

Aggregation	Add	Mean	Max
GraphSAGE Positioning Error	6.9749	6.7011	6.7642
GraphConv Positioning Error	6.8658	6.6571	6.6759

ISSN 2723-6471

4.4. Hidden Channels Tuning

In table 10 we evaluate the effect of different numbers of hidden channels on GAT, GraphSAGE, and GraphConv. The optimal hidden channels found for GAT model are 64 channels achieving a positioning error of 6.6609. It is a significant improvement from 32 hidden channels. However further increases in channel numbers further degrade performance.

Table 10. Hidden Channels Fine-tuning Results

Hidden Channels	32	64	128	256
GAT Positioning Error	6.9776	6.6609	6.8001	6.9378
GraphSAGE Positioning Error	7.0631	6.6571	6.5600	6.5612
GraphConv Positioning Error	7.0271	6.6571	6.5268	6.5362

Similarly, GraphSAGE and GraphConv perform optimally with 128 hidden channels achieving a positioning error of 6.5600 and 6.5268 respectively. That is a significant improvement from 32 and 64 hidden channels but further increases in the number of hidden channels do not appear to provide any benefit. In general, the results indicate that increases of hidden channels worked to improve the model up to a certain point as it can increase information extraction and provide more representation for the data. However, further increases from that point reduces performance as it introduces too much unnecessary model complexity.

4.5. Activation Layer Tuning

In table 11 we experimented with different negative slopes for the LeakyReLU activation function. The experiment showed that a negative slope of 0.1 consistently produced the best results across all model architectures (6.5893 for GAT, 6.3349 for GraphSAGE, and 6.3278 for GraphConv). This suggests that smaller negative slope values better strike the balance between addressing the vanishing gradient problems, while having relatively low negative impact on model quality.

Table 11. Activation Layer Fine-tuning Results

Negative Slope	0.1	0.2	0.3	0.4
GAT Positioning Error	6.5893	6.6609	7.0178	6.9218
GraphSAGE Positioning Error	6.3349	6.5600	6.6858	6.7331
GraphConv Positioning Error	6.3278	6.5268	6.6359	6.8856

4.6. Dropout Layer Tuning

Table 12 analyze the effect of different dropout rates on each model architecture. The best performance for GAT is achieved with a small dropout rate of 0.1 (6.5440). Higher dropout rates seem to negatively impact performance, possibly because GAT already applies strong attention-based feature selection. In contrast, both GraphSAGE and GraphConv prefer a larger dropout rate of 0.4, achieving positioning errors of 6.2246 and 6.2931 respectively. These results suggest that these architectures may benefit from regularization at a higher rate to prevent overfitting. Once again, we found that GraphSAGE and GraphConv prefers similar model parameters. Meanwhile GAT, a more radical departure from the classic GNN model, prefers a different configuration to the other two architectures.

Table 12. GAT Dropout Layer Fine-tuning Results

Dropout Rate	0.1	0.2	0.3	0.4	0.5
GAT Positioning Error	6.5440	6.5879	6.5893	6.5888	6.7327
GraphSAGE Positioning Error	6.4441	6.3304	6.3349	6.2246	6.3178
GraphConv Positioning Error	6.4107	6.3101	6.3278	6.2931	6.3260

 ISSN 2723-6471

4.7. Optimizer and Learning Rate Tuning

In the final stage of our fine-tuning process, we explored the impact of different optimizers and learning rates on model performance, as presented in Figure 7. We found that in general, models trained via transfer learning schema B perform better compared to the models trained via schema A and significantly outperform traditional non transfer learning methods. This result highlights the effectiveness of transfer learning in enhancing the capabilities of the model when applied to indoor positioning tasks, pertinently when applied to specialized floor-based models.

Figure 7. Optimizer and Learning Rate Fine-tuning Results

From the results, we can see that in general Adam and its newer variant AdamW performs better than RMSProp in every model architecture. AdamW seems to do better on the Schema B model, whilst AdamW extracts greater result out of Schema A and the Unified (Non-TF) model. For GAT, Adam with a learning rate of 0.01 under Schema B achieves the lowest positioning error (6.5440), whereas RMSProp performs worse across all learning rates, particularly at higher values where the error increases significantly. A similar pattern is observed in GraphSAGE, where Adam achieves the lowest positioning error (6.2246) at a learning rate of 0.01 under Schema B, while RMSProp struggles to maintain stability, particularly at higher learning rates.

For GraphConv the performance gap between optimizers is even more pronounced. The best overall result is achieved by GraphConv with Adam at a 0.01 learning rate under Schema B, reaching a positioning error of just 6.2176, which is the lowest error recorded across all model configurations. From our testing, it seems that GraphConv performs the best, closely followed by GraphSAGE, then GAT at last place. The learning rates preferred by each configuration seem to be varied as each model architecture and training schema performs best under different learning rates.

As can be seen from the comparison on table 23, our model managed to produce results with significantly smaller positioning errors than the original model created by [6] for UTSIndoorLoc dataset. After conducting several more runs, we found that the model has a worse mean of 6.946 with a standard deviation of 0.12. Whilst this result is still noticeably worse than the best run, even this is still significantly better than the CNNLoc model. This result illustrates the efficacy of transfer learning in graph-based neural networks for indoor positioning within the specified dataset. However, given that the evaluation is confined to a single dataset and lacks cross-dataset validation or ablation analysis, there still needs to be more research regarding the generalizability of this methodology to confidently state that this model is superior to CNNLoc.

Table 23. GraphConv Dropout Layer Fine-tuning Results

Models	Mean Positioning Error
GraphConv Schema B (Best)	6.2176 m
GraphConv Schema B (Mean ± σ)	6.3946 ± 0.12 m
CNNLoc [6]	7.60 m

5. Conclusion

Our research focused on evaluating the impact of transfer learning on the performance of Graph Attention Networks (GAT), GraphSAGE, and Graph Convolutional Networks (GraphConv) for indoor positioning tasks. In it we devised two transfer learning training schemas in order to achieve better results than traditional model training. Based on our findings, we concluded GraphConv model with schema B training managed to achieve the lowest positioning error in our experiments. This model considerably improves upon the original model created by [6] for UTSIndoorLoc dataset achieving a 6.2176 m positioning error compared to 7.60 m in their research. However, it is important to acknowledge that the reported performance is done in a single dataset and lacks cross-dataset validation or ablation analysis, therefore needing further experiments to confidently conclude that it is an overall superior methodology.

In the future, we would like to see this experiment to be replicated on other indoor positioning datasets as this dataset has several problems such as duplicated data and unbalanced data distribution per floor. Aside from that, the fine-tuning process can still be improved now that we know that GraphConv performs the best given that the data and graphs related fine-tuning were done using GAT model as reference. Finally, we would like to see an actual application of this model as this research so far has only produced research models and hasn't seen real world implementations yet

6. Declarations

6.1. Acknowledgements

The authors wish to convey heartfelt appreciation to Justin Orlean, a senior student in the Technology Information program at Binus University, for their invaluable support in reviewing and rectifying the code utilized in this research. Their insights and recommendations significantly enhanced the implementation and contributed to the precision of the experimental outcomes.

6.2. Author Contributions

Conceptualization: K.W., H.M.S.B., and G.P.K.; Methodology: K.W., H.M.S.B., and G.P.K.; Software: K.W. and H.M.S.B.; Validation: K.W. and H.M.S.B.; Formal Analysis: K.W. and H.M.S.B.; Resources: K.W. and H.M.S.B.; Data Curation: K.W. and G.P.K.; Writing Original Draft Preparation: K.W. and H.M.S.B.; Writing Review and Editing: G.P.K.; Visualization: K.W.; All authors have read and agreed to the published version of the manuscript.

6.3. Data Availability Statement

The data presented in this study are available on request from the corresponding author. The dataset used in this study was originally published by Song et al. in "A Novel Convolutional Neural Network Based Indoor Localization Framework with WiFi Fingerprinting" [6].

6.4. Funding

No funding was received for the research and authorship of this article. Partial reimbursement was provided solely to cover publication-related expenses. This support had no role in the study design, experimentation, analysis, interpretation of results, or manuscript preparation.

6.5. Institutional Review Board Statement

Not applicable.

6.6. Informed Consent Statement

Not applicable.

6.7. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] E. A. Maguire, N. Burgess, J. G. Donnett, R. S. J. Frackowiak, C. D. Frith, and J. O'Keefe, "Knowing Where and Getting There: A Human Navigation Network," *Science (1979)*, vol. 280, no. 5365, pp. 921–924, May 1998, doi: 10.1126/science.280.5365.921.

[2] S. M. Asaad and H. S. Maghdid, "A Comprehensive Review of Indoor/Outdoor Localization Solutions in IoT era: Research Challenges and Future Perspectives," *Computer Networks*, vol. 212, p. 109041, Jul. 2022, doi: 10.1016/j.comnet.2022.109041.

[3] S.-G. Martensson, "Height Determination by GPS - Accuracy with Respect to Different Geoid Models in Sweden," *Proceedings of the XXII FIG International Congress, Washington, D.C., USA.*, Jan. 2002.

[4] Y. Hu *et al.*, "Experience: practical indoor localization for malls," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, New York, NY, USA: ACM, Oct. 2022, pp. 82–93. doi: 10.1145/3495243.3517021.

[5] B. Tan *et al.*, "Improved Sensing and Positioning via 5G and mmWave radar for Airport Surveillance," Feb. 2022.

[6] X. Song *et al.*, "A Novel Convolutional Neural Network Based Indoor Localization Framework With WiFi Fingerprinting," *IEEE Access*, vol. 7, pp. 110698–110709, 2019, doi: 10.1109/ACCESS.2019.2933921.

[7] A. Riady and G. P. Kusuma, "Indoor positioning system using hybrid method of fingerprinting and pedestrian dead reckoning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7101–7110, Oct. 2022, doi: 10.1016/j.jksuci.2021.09.005.

[8] L. Bai, F. Ciravegna, R. Bond, and M. Mulvenna, "A Low Cost Indoor Positioning System Using Bluetooth Low Energy," *IEEE Access*, vol. 8, pp. 136858–136871, 2020, doi: 10.1109/ACCESS.2020.3012342.

[9] Y. Fu, X. Xiong, Z. Liu, X. Chen, Y. Liu, and Z. Fu, "A GNN-based indoor localization method using mobile RFID platform," in *2022 7th International Conference on Smart and Sustainable Technologies (SpliTech)*, IEEE, Jul. 2022, pp. 1–6. doi: 10.23919/SpliTech55088.2022.9854370.

[10] H. Alkan and H. Celebi, "The Implementation of Positioning System with Trilateration of Haversine Distance," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, IEEE, Sep. 2019, pp. 1–6. doi: 10.1109/PIMRC.2019.8904289.

[11] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-Based Indoor Localization with Bluetooth Low Energy Beacons," *Sensors*, vol. 16, no. 5, p. 596, Apr. 2016, doi: 10.3390/s16050596.

[12] S. Shang and L. Wang, "Overview of WiFi fingerprinting-based indoor positioning," *IET Communications*, vol. 16, no. 7, pp. 725–733, Apr. 2022, doi: 10.1049/cmu2.12386.

[13] W. Honcharenko, H. L. Bertoni, and J. Dailing, "Mechanisms governing propagation between different floors in buildings," *IEEE Trans Antennas Propag*, vol. 41, no. 6, pp. 787–790, Jun. 1993, doi: 10.1109/8.250441.

[14] F. Lezama, G. G. Gonzalez, F. Larroca, and G. Capdehourat, "Indoor Localization using Graph Neural Networks," in *2021 IEEE URUCON*, IEEE, Nov. 2021, pp. 51–54. doi: 10.1109/URUCON53396.2021.9647082.

[15] F. Gu *et al.*, "Accurate and efficient floor localization with scalable spiking graph neural networks," *Satellite Navigation*, vol. 5, no. 1, p. 6, Dec. 2024, doi: 10.1186/s43020-024-00127-8.

[16] X. Kang, X. Liang, and Q. Liang, "Indoor Localization Algorithm Based on a High-Order Graph Neural Network," *Sensors*, vol. 23, no. 19, p. 8221, Oct. 2023, doi: 10.3390/s23198221.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," Sep. 2016.

[18] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," Jun. 2017.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," Oct. 2017.

[20] X. Han, Z. Huang, B. An, and J. Bai, "Adaptive Transfer Learning on Graph Neural Networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Jul. 2021, pp. 565–574.

[21] M. Zhang, Z. Fan, R. Shibasaki, and X. Song, "Domain Adversarial Graph Convolutional Network Based on RSSI and Crowdsensing for Indoor Localization," *IEEE Internet Things J*, vol. 10, no. 15, pp. 13662–13672, Aug. 2023, doi: 10.1109/JIOT.2023.3262740.

[22] Y. Dong, T. Arslan, and Y. Yang, "A Multimodal Graph Fingerprinting Method for Indoor Positioning Systems," in *2023 13th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, Sep. 2023, pp. 1–6. doi: 10.1109/IPIN57070.2023.10332496.

[23] R. Vishwakarma, R. B. Joshi, and S. Mishra, "IndoorGNN: A Graph Neural Network Based Approach for Indoor Localization Using WiFi RSSI," 2023, pp. 150–165. doi: 10.1007/978-3-031-49601-1_11.

[24] X. Luo and N. Meratnia, "A Geometric Deep Learning Framework for Accurate Indoor Localization," in *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, Sep. 2022, pp. 1–8. doi: 10.1109/IPIN54987.2022.9918118.

[25] M. Kim, "Graph-Based Machine Learning for Practical Indoor Localization," *IEEE Sens Lett*, vol. 6, no. 12, pp. 1–4, Dec. 2022, doi: 10.1109/LSENS.2022.3224818.

[26] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," *J Big Data*, vol. 9, no. 1, p. 102, Oct. 2022, doi: 10.1186/s40537-022-00652-w.

[27] S. Kang, "k-nearest neighbor learning with graph neural networks," *Mathematics*, vol. 9, no. 8, p. 830, Apr. 2021, doi: 10.3390/math9080830.

[28] I. Neupane, S. Shahrestani, and C. Ruan, "Indoor Localization of Resource-Constrained IoT Devices Using Wi-Fi Fingerprinting and Convolutional Neural Network," in *Proceedings of the 2024 Australasian Computer Science Week*, New York, NY, USA: ACM, Jan. 2024, pp. 20–25. doi: 10.1145/3641142.3641158.

[29] S. He and S.-H. G. Chan, "Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 466–490, 2016, doi: 10.1109/COMST.2015.2464084.

[30] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Trans Neural Netw*, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/TNN.2008.2005605.

[31] J. Xu, Z. Li, B. Du, M. Zhang, and J. Liu, "Reluplex made more practical: Leaky ReLU," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, Jul. 2020, pp. 1–7. doi: 10.1109/ISCC50000.2020.9219587.

[32] J. Torres-Sospedra, R. Montoliu, S. Trilles, Ó. Belmonte, and J. Huerta, "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems," *Expert Syst Appl*, vol. 42, no. 23, pp. 9263–9278, Dec. 2015, doi: 10.1016/j.eswa.2015.08.013.