Indoor Localization in IoT Networks Based on Graph Neural Networks

GUO Xiaofan

23 / 01 / 2025

Table of Contents

1. Background of Indoor Localization

1.1 Importance and Challenges

- ➢ Growing demand for indoor localization in malls, hospitals, etc.
- ➢ Current systems lack accuracy and reliability, limiting potential.

1.2 Objective

- ➢ Use Deep Learning to improve indoor localization accuracy.
- ➢ Focus predicting the following within a single building:
    - Longitude
    - Latitude
    - Floor

2. Database: UJIIndoorLoc

2.1 Advantages of the UJIIndoorLoc Database

- Largest public indoor localization database for algorithm benchmarking.
- **Key features:**
  - Multi-building, multi-floor coverage.
  - Diverse and large sample set.
  - Open access with WLAN fingerprinting data.

2.2 Data Used in This Project

- **RSSI data**: First 519 columns.
- **Geographic information**: Columns 520-522 (Longitude, Latitude, Floor).

3. Deep Neural Networks (DNN)

3.1 Advantages of DNN

- Handles high-dimensional data efficiently.
- Extracts non-linear features, suitable for indoor localization.

3.2 DNN Structure

- **Input Layer:** Processes 519-dimensional RSSI data.
- **Encoding Layers:** Two layers (256 & 64 neurons) & ELU activation.
- **Decoding Layers:** Reconstructs input features.
- **Output Layer:** Generates compressed features for classification or regression.

4. Graph Neural Networks (GNN)

4.1 Advantages of GNN

- Models RSSI data as a graph, capturing spatial and topological relationships.
- Combines local and global data features using message-passing mechanisms.

4.2 GNN Structure

$$d(i,j) = \sqrt{\sum_{k=1}^{n}\left(x_{i,k} - x_{j,k}\right)^2}$$

- **Data Representation:** Each sample is a node; connections are defined using k-nearest neighbors (k-NN), uses Euclidean distance to calculate similarity.
- **Message Passing:** Nodes exchange and aggregate information from neighbors.
- **Graph Convolutional Layers (GCN):** uses ELU activation in graph convolutional layers to aggregate node features via a normalized adjacency matrix, capturing broader neighborhood information with each layer.

$$X^{(l+1)} = \sigma\left(\hat{A}X^{(l)}W^{(l)}\right)$$

- **Output Layer:** Supports classification or regression tasks on RSSI data.

5. Data Processing and Implementation

5.1 Data Preprocessing

- **Activation Function:** ELU is used due to RSSI data being negative, ensuring better gradient stability and faster convergence.

$$ELU(x) = \begin{cases} x, & x > 0 \\ \alpha(e^{x} - 1), & x \leq 0 \end{cases}$$

- **Loss Function**: MSE minimizes differences between predicted and true longitude/latitude, improving accuracy.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- **Data Normalization**: MinMaxScaler scales RSSI, longitude, and latitude data to [0, 1]. Ensures consistent feature scaling and reduces training complexity.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- **Data Denormalization:** Restores predicted values (longitude, latitude) to original scales for real-world relevance.

$$X_{original} = X_{norm} \cdot (X_{max} - X_{min}) + X_{min}$$

5. Data Processing and Implementation

5.2 Training Process

- **Optimizer:**
  - Adam dynamically adjusts learning rates for faster and smoother convergence.
- **Model Parameters:**
  - **Input dimensions**: 519 (RSSI features).
  - **Dropout rate**: 0,5 (DNN) or 0,1 (GNN)
  - **Epochs**: Tuned based on performance to prevent overfitting.
  - **Batch size**: Adjusted for efficient memory use.
  - **K of K-NN :** The number of nearest neighbors considered for each node during graph construction.

5. Data Processing and Implementation

5.3 Error Detection

- **Purpose:** Evaluates model accuracy by calculating the deviation between predicted and actual longitude, latitude, and floor values.
- **Error Metric:** Root Mean Square Error (RMSE) is used for precise measurement.
- **floor_penalty:** prioritizes floor accuracy in localization (floor_penalty =4).
- **Result:** Outputs Average Error (meters), combining geographic and floor accuracy for comprehensive evaluation.

$$Longitude\ Error = \sqrt{\left(\hat{y}_{lng} - y_{lng}\right)^2}$$

$$Latitude\ Error = \sqrt{(\hat{y}_{lat} - y_{lat})^2}$$

$$Floor\ Error = \sqrt{\left(\hat{y}_f - y_f\right)^2}$$

$$Floor\ Error\ Sum = \sum Floor\ Error$$

$$Coordinates\ Error\ Sum = \sum (Longitude\ Error + Latitude\ Error)$$

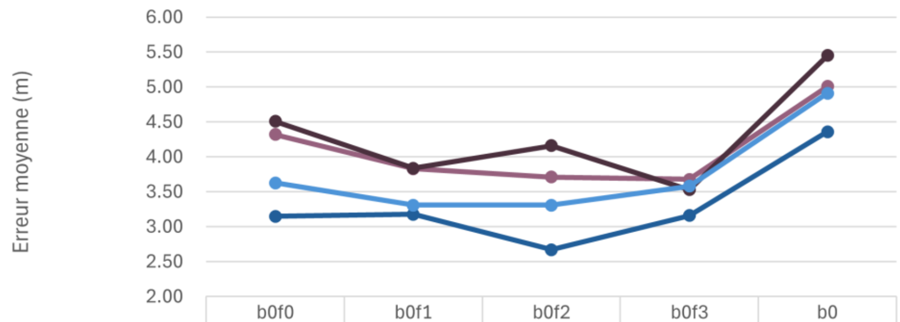
$$Total\ Error = floor\_penalty \times Floor\ Error\ Sum + Coordinates\ Error\ Sum$$

$$Average\ Error = \frac{Total\ Error}{N}$$

6. Results Evaluation

Le = Longitude epochs = Latitude epochs
BS = Batch size

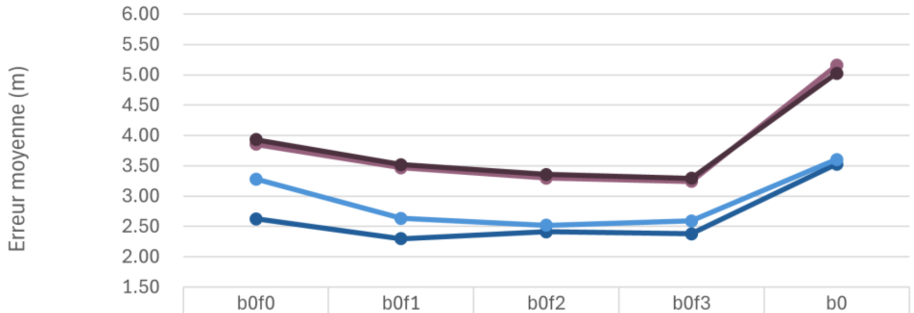
DNN-Données d'Entraînement (TrainingData)



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,BS=32	4.32	3.83	3.71	3.68	5.01
Le=1000,Fe=500,BS=128	4.51	3.84	4.16	3.53	5.45
Le=3000,Fe=1000,BS=32	3.15	3.18	2.67	3.16	4.36
Le=3000,Fe=1000,BS=128	3.63	3.31	3.31	3.58	4.91

Étage du Bâtiment 0

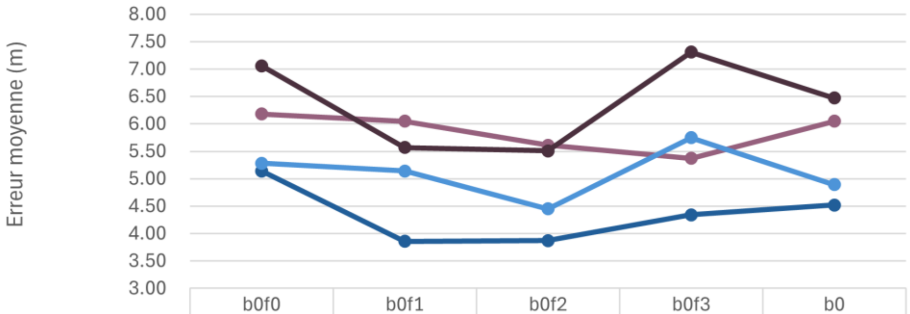
Données d'Entraînement (TrainingData)



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,K=5	3.85	3.47	3.30	3.24	5.16
Le=1000,Fe=500,K=15	3.93	3.52	3.35	3.30	5.02
Le=3000,Fe=1000,K=5	2.62	2.30	2.41	2.38	3.53
Le=3000,Fe=1000,K=15	3.28	2.64	2.52	2.59	3.60

Étage du Bâtiment 0

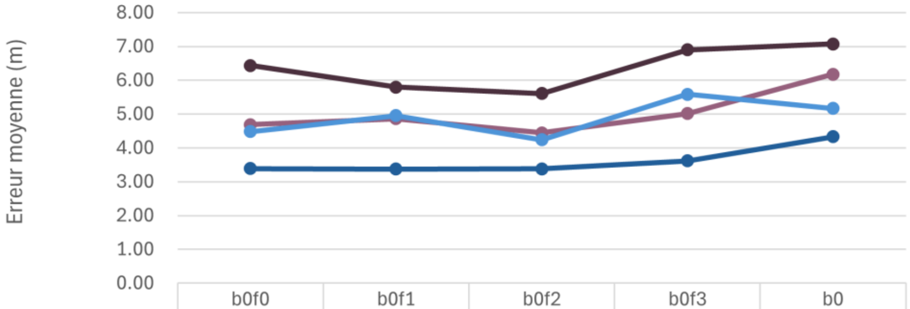
DNN-Données de Test (TestData)



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,BS=32	6.18	6.05	5.61	5.37	6.05
Le=1000,Fe=500,BS=64	7.06	5.57	5.51	7.31	6.47
Le=3000,Fe=1000,BS=32	5.14	3.86	3.87	4.34	4.52
Le=3000,Fe=1000,BS=64	5.28	5.14	4.45	5.75	4.89

Étage du Bâtiment 0

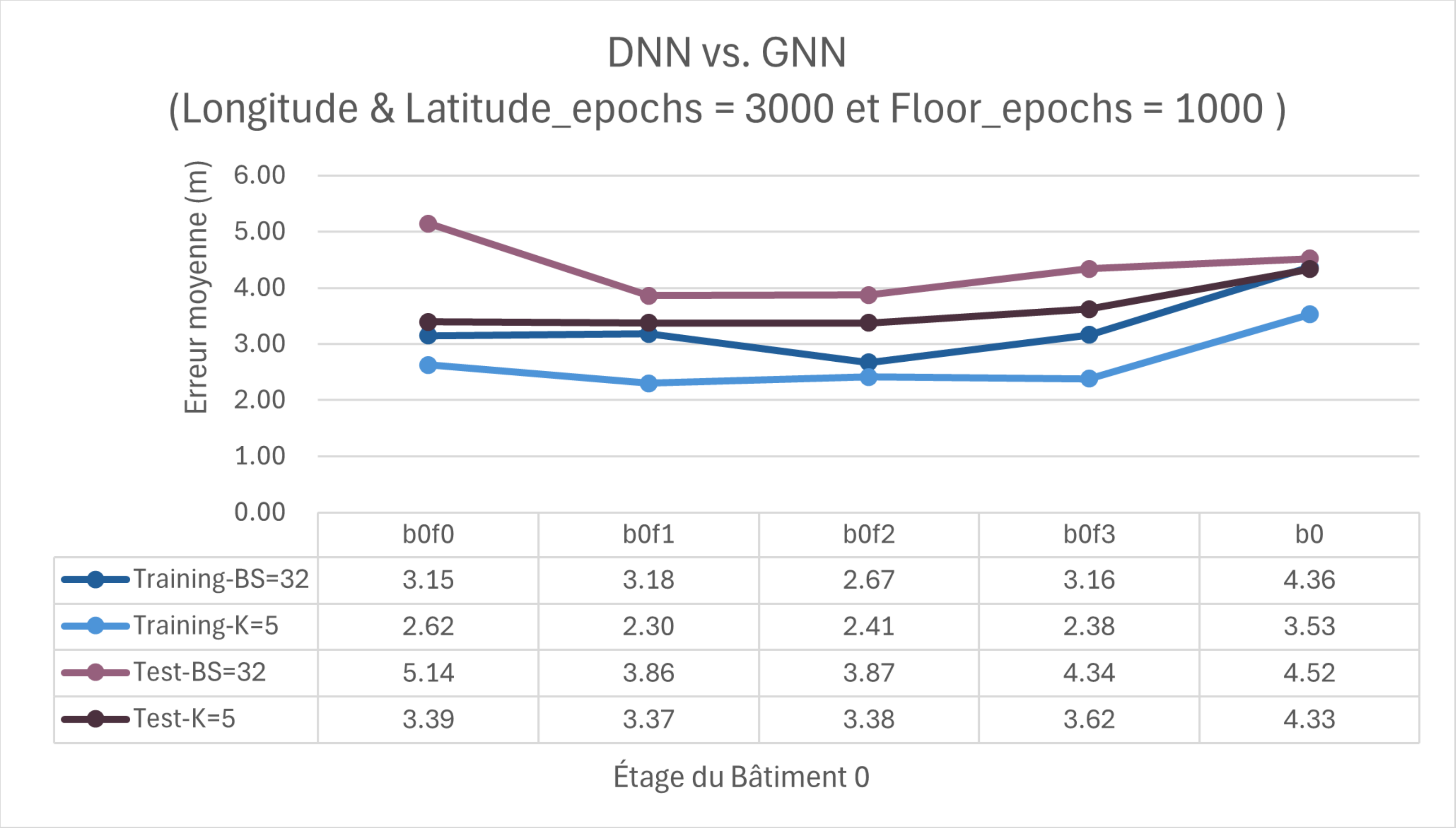
GNN-Données de Test (TestData)



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,K=5	4.69	4.87	4.44	5.01	6.18
Le=1000,Fe=500,K=15	6.43	5.80	5.61	6.90	7.08
Le=3000,Fe=1000,K=5	3.39	3.37	3.38	3.62	4.33
Le=3000,Fe=1000,K=15	4.48	4.95	4.24	5.58	5.17

Étage du Bâtiment 0

6. Results Evaluation



	b0f0	b0f1	b0f2	b0f3	b0
Training-BS=32	3.15	3.18	2.67	3.16	4.36
Training-K=5	2.62	2.30	2.41	2.38	3.53
Test-BS=32	5.14	3.86	3.87	4.34	4.52
Test-K=5	3.39	3.37	3.38	3.62	4.33

- GNN (K=5) achieves lower average errors than DNN (BS=32) across all floors.
- Training error for GNN is consistently smaller, especially on b0f0 and b0f1.
- Testing error with GNN shows notable improvement, reducing large deviations seen in DNN.
- GNN effectively captures spatial relationships, ensuring higher localization accuracy.

7. Complexity

DNN Model Complexity:

```
--- Model Complexity ---
Encoder Model: Parameters = 299591, FLOPs = 299591
Longitude Model: Parameters = 223937, FLOPs = 223937
Latitude Model: Parameters = 223937, FLOPs = 223937
Floor Model: Parameters = 199233, FLOPs = 199233

Overall Model Complexity: Parameters = 946698, FLOPs = 946698
DNN Complexity: Parameters = 946698, FLOPs = 946698
Finish
```

GNN Model Complexity:

```
--- Model Complexity ---
Longitude Model: Parameters = 264961, FLOPs = 5530625
Latitude Model: Parameters = 264961, FLOPs = 5530625
Floor Model: Parameters = 264961, FLOPs = 5530625

Overall Model Complexity: Parameters = 794883, FLOPs = 16591875
Finish.
```

- **Model Complexity:** GNN has higher computational complexity than DNN due to graph construction and message passing.
- **Data Representation:** Transforming data into graph structures (using k-NN) adds preprocessing overhead.
- **Training Time:** GNN requires more time per epoch as it aggregates information from neighboring nodes.
- **Trade-off:** The increased complexity of GNN results in better accuracy and improved spatial understanding.

8. Conclusion

8.1 Comparison of DNN and GNN

- **DNNs:** Simple, fast, suitable for unstructured data but limited for complex relationships.
- **GNNs:** Better accuracy for spatial and topological tasks but higher computational cost.

8.2 Key Findings

- GNNs outperform DNNs in tasks requiring relational understanding like indoor localization.

8.3 Future Directions

- Optimize model parameters.
- Explore hybrid DNN-GNN approaches.
- Integrate inertial sensor data for better performance.